Jaume Jurado Jan Izquierdo
Joan Lloret David Marquez

ENERGY ANALYSIS EXERCISE

Project description:

The main objective of this project is to obtain the relative contribution of every residue in terms of energy given a protein protein interaction. For this project we will be evaluating the energies of the complex RBD-ACE2, found in PDB as 6m0j.
To get to this goal, we first have to retrieve the structure from PDB and refine it in order to be ready to work with it. After this, we will look for all the residues that are close enough to each other to consider that an interaction between them is possible. Once this residues are properly identified and stored we will exchange each residue with an alanine, that is the least energetic amino acid and compute the difference of energy with the original residue. After this we will know how much energy the residues in the protein protein interaction poses.

Processes code description:

In order to ensure a correct functioning of the code first we have to obtain and refine the data we will use, that is the .pdb file that stores the protein we want to use. Apart from these preparation steps we also will have to include some modules of the biopython package, specifically the Bio.PDB.NeighborSearch and the Bio.PDB.PDBParser.

Preparation steps

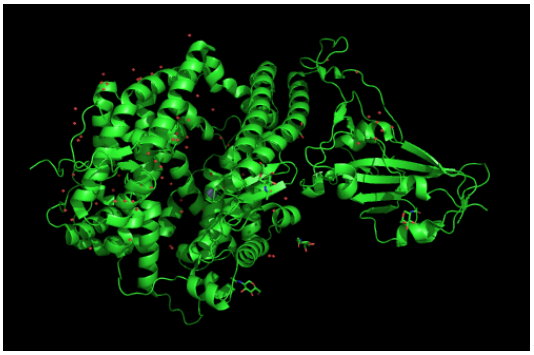
In the preparation steps the structure of 6m0j is retrieved from PDB by using the parser.get_structure() and stored in a variable.

Then, using the function args it is checked at PDB the composition of a biological unit and all chains but those involved in the biological unit are removed.

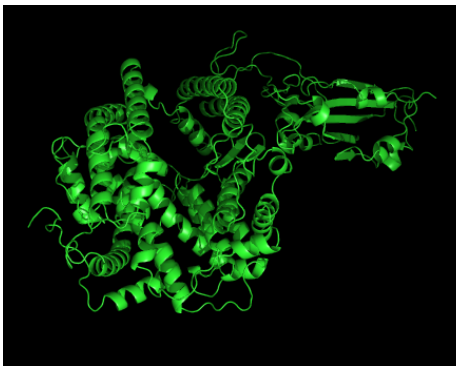
After this, the next step is the cleansing and checking of the structure, this is done with the StructureChecking function and args function. More precisely, this step allows us to remove all heteroatoms, fix the amides, the chirality and the backbone, detect and rebuild if there are missing protein chains and add the necessary hydrogens.

After all this, the desired structure is correctly imported, stored and refined.

Here is a comparison between the protein structure before and after the cleansing and refining:



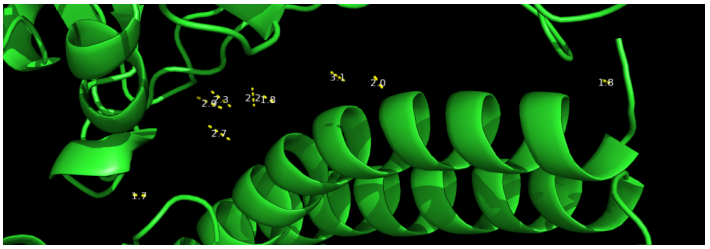
Before the preparation steps



After the preparation steps

Step 1:

In this step we had to both determine a suitable distance in the way that all contact residues are included, and add 1 or 2 A so the adjacent residues are also considered and then prepare a python script that defines a list of interface residues on each chain.
Firstly we visually inspected our molecule on PyMol and used the action menu → find → polar contacts → between chains. With this, the program itself selects which contacts should be considered, and we take the higher value, so all the rest will be included, as the reference value.



As can be seen in this image, the highest value we will consider is 4, that is the biggest distance in the image + 1 Å to ensure that the neighbours are taken into account.
Then, as per the python script, that can be found in the step1_2.py, we defined two functions, the first one, called chain_atoms() stores the model in a variable and then separates it into its different chains, for then storing the separated chains in different variables. The second function called chain_comparison() takes as input both chains

stored in separated functions and the distance threshold we chose, then it computes the distance between every residue in the chains and compares this distance to the threshold value, if the distance is lower than the threshold value, the residues are stored in a list that then is returned.

Step 2:

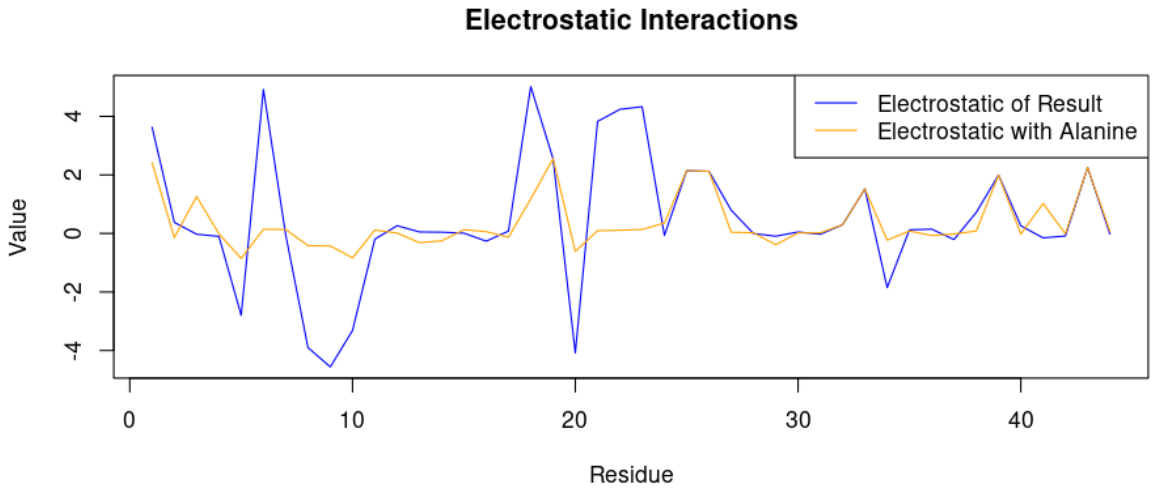
For this step we first use a structure similar to step one for obtaining in a list the desired atoms of every chain and its neighbours in different variables. Once we have two variables, each of them storing the relevant atoms of every chain, we execute the second function. What the second function does is a calculation of all the electrostatic interactions, Van der Waals interactions, Solvation of the A-B complex, Solvation of A independently, solvation of B independently and the interaction energy between components in A-B complex.
All of this is done using auxiliary functions found in the S2m3_module2.py file.

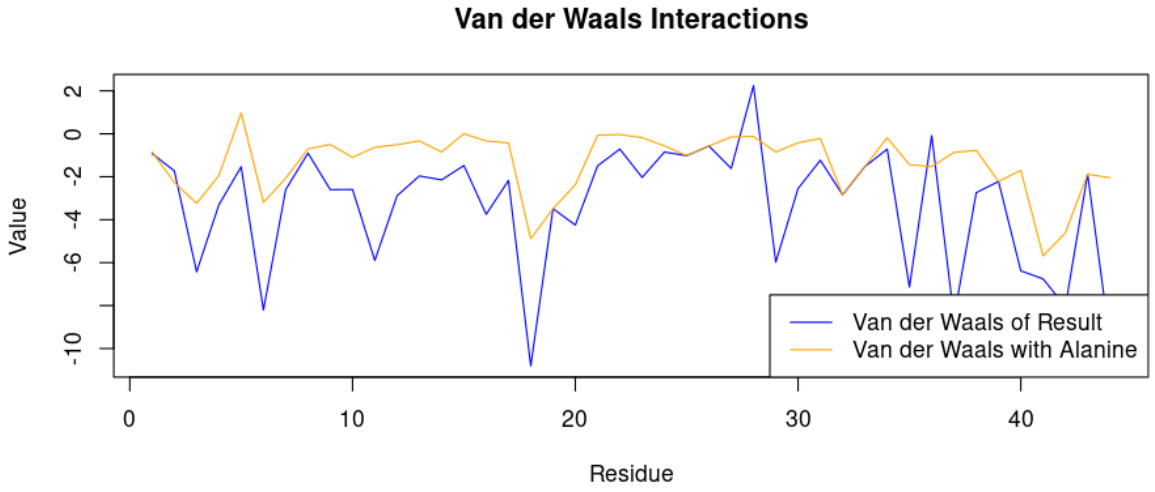
Step 3:

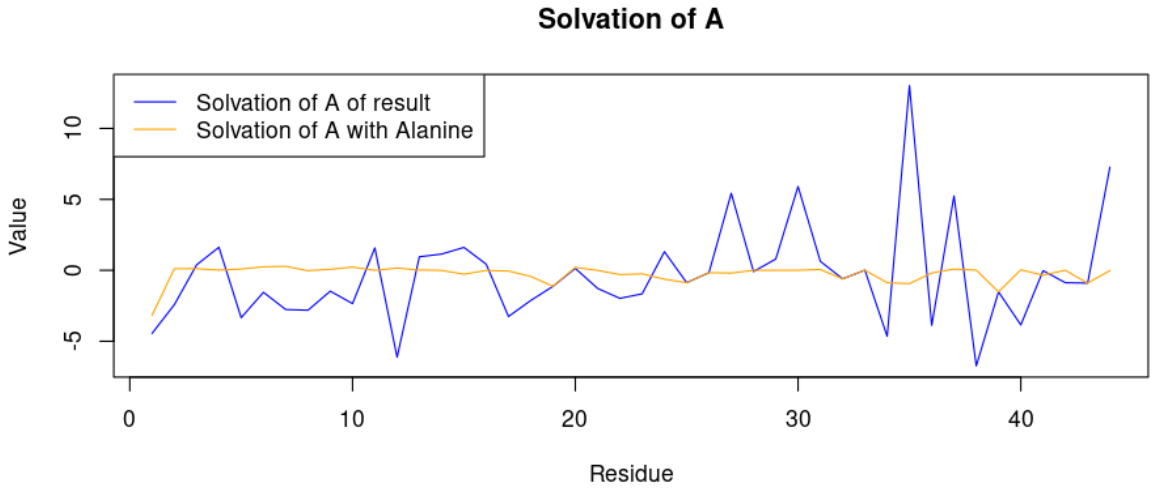
For this step we do the same as in step two but analysing the energy of every residue exchanged with an alanine, which is the amino acid with less energy as it is the most simple, then we have store all the values obtained in a csv file, that also contains all energies of the natural residue and of the alanine, this way we can compare it using an R script.
This is how the csv outputs:

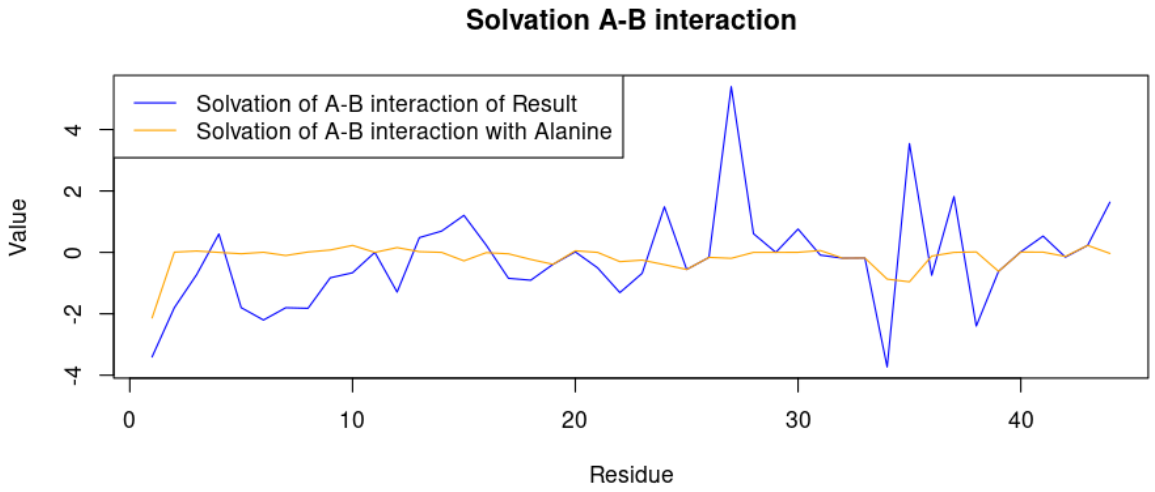
	A	B	C	D	E	F	G	H	I	J
1	D#res_id	elec_res	vdw_res	solv_AB_res	solv_A_res	-	elec_ala	vdw_ala	solv_AB_ala	solv_A_ala
2	D#SER A19	3.6318	-0.9238	-3.4036	-4.4436	-	2.4167	-0.8400	-2.1335	-3.1735
3	D#GLN A24	0.3767	-1.7267	-1.8006	-2.4184	-	-0.1379	-2.2663	0.0075	0.1122
4	D#THR A27	-0.0266	-6.4291	-0.7248	0.3872	-	1.2623	-3.2416	0.0444	0.1200
5	D#PHE A28	-0.0999	-3.3071	0.5986	1.6252	-	-0.0112	-1.9429	-0.0005	0.0182
6	D#ASP A30	-2.7963	-1.5358	-1.8000	-3.3357	-	-0.8539	0.9669	-0.0448	0.0908
7	D#LYS A31	4.9173	-8.2069	-2.2030	-1.5507	-	0.1407	-3.1928	0.0038	0.2487
8	D#HIE A34	-0.0567	-2.5958	-1.8016	-2.7625	-	0.1350	-2.0718	-0.1024	0.2772
9	D#GLU A35	-3.9023	-0.9008	-1.8210	-2.8108	-	-0.4196	-0.7045	0.0124	-0.0230
10	D#GLU A37	-4.5615	-2.6018	-0.8279	-1.4683	-	-0.4267	-0.5031	0.0775	0.0775
11	D#ASP A38	-3.3145	-2.5945	-0.6635	-2.3447	-	-0.8360	-1.0974	0.2275	0.2275
12	D#TYR A41	-0.1969	-5.8972	0.0000	1.5778	-	0.1175	-0.6276	0.0000	0.0000
13	D#GLN A42	0.2662	-2.8827	-1.2924	-6.1210	-	0.0139	-0.5063	0.1565	0.1565
14	D#LEU A45	0.0514	-1.9649	0.4812	0.9568	-	-0.3141	-0.3286	0.0229	0.0229
15	D#LEU A79	0.0426	-2.1463	0.6936	1.1452	-	-0.2538	-0.8501	0.0000	-0.0060
16	D#MET A82	0.0081	-1.4779	1.2067	1.6147	-	0.1289	0.0026	-0.2765	-0.2695
17	D#TYR A83	-0.2630	-3.7501	0.2354	0.4332	-	0.0602	-0.3334	-0.0091	-0.0091
18	D#ASN A330	0.0794	-2.1726	-0.8447	-3.2533	-	-0.1332	-0.4241	-0.0466	-0.0466

With this file we then can plot the data and for each type of energy see the comparison between the original residue and the alanine, this are the graphs that we obtained:


Electrostatic Interactions
Electrostatic of Result
Electrostatic with Alanine
Value
Residue


Van der Waals Interactions
Van der Waals of Result
Van der Waals with Alanine
Value
Residue


Solvation of A
Solvation of A of result
Solvation of A with Alanine
Value
Residue

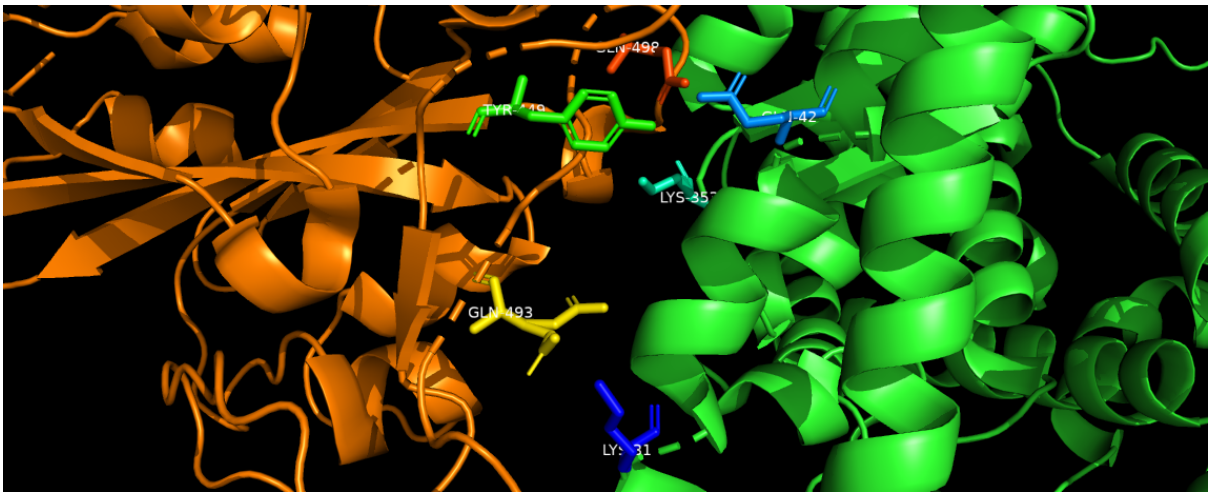


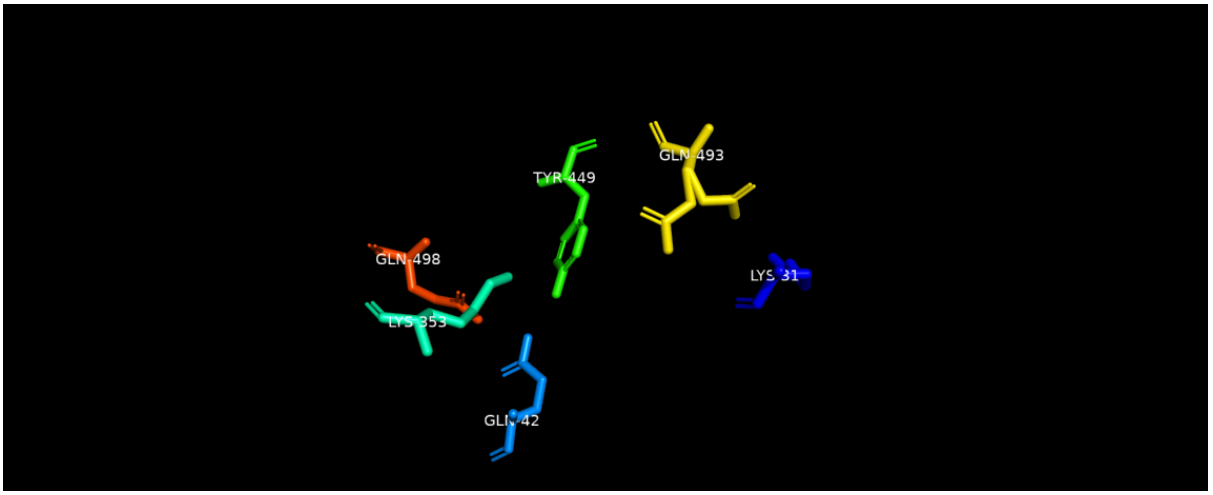
Step 4:

In this step we are asked to take images of the residues we consider are the most important. In order to do this we look at the peaks in the plots of step 3 which are the residues that suffer the most energy change when doing the Alanine exchange.
We separated these residues according to their importance for each interaction, the criteria was the difference in the graphs. The methods used to produce the images were recoloring chain A to be green and chain E to be orange,identifying the residues and changing each residue to a different colour, and hiding everything but these residues

Electrostatic

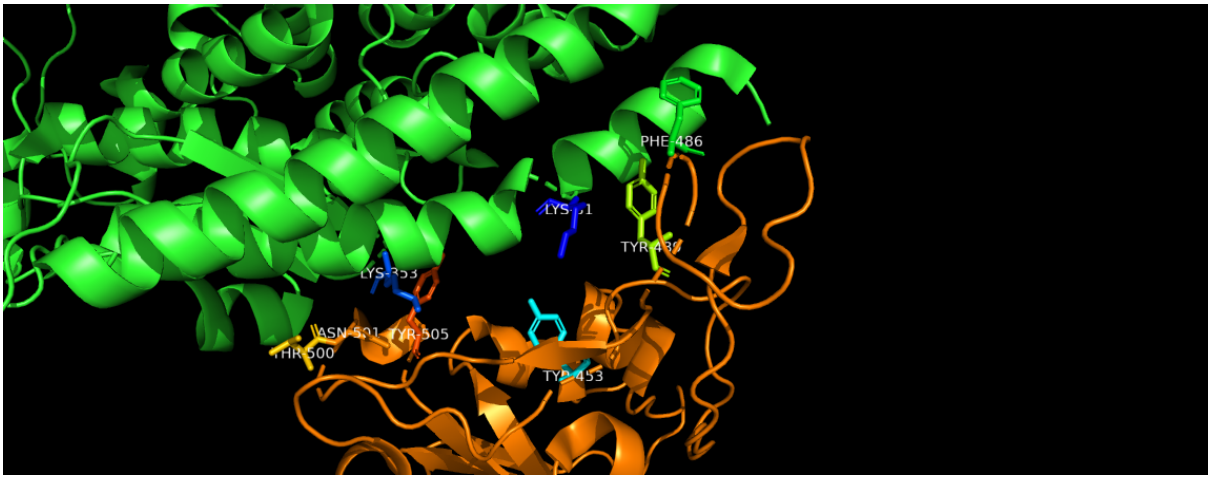
LYS A31, GLN A42, LYS A353, TYR E449, GLN E493 and , GLN E498.

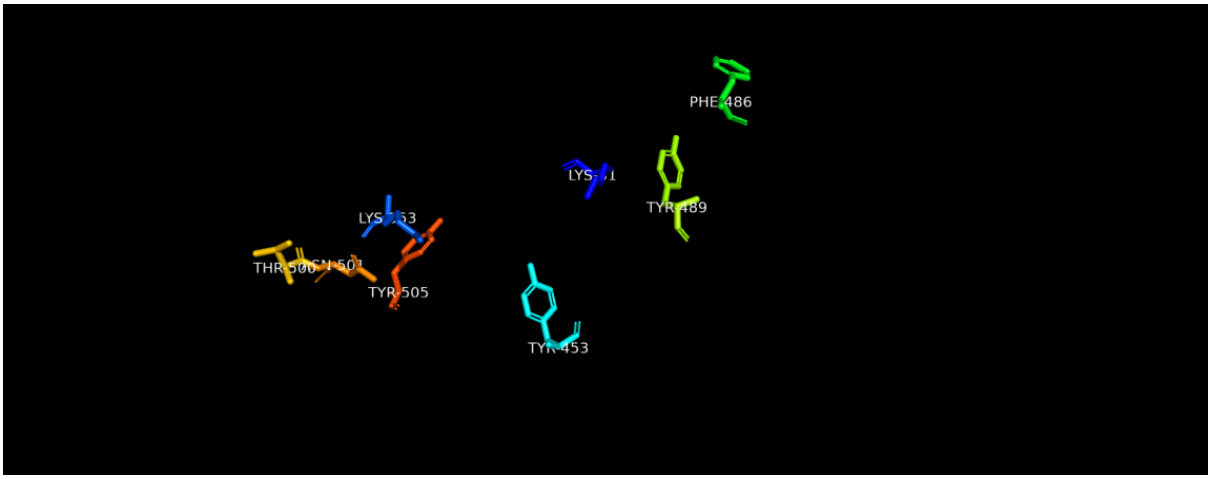




Van der Waals

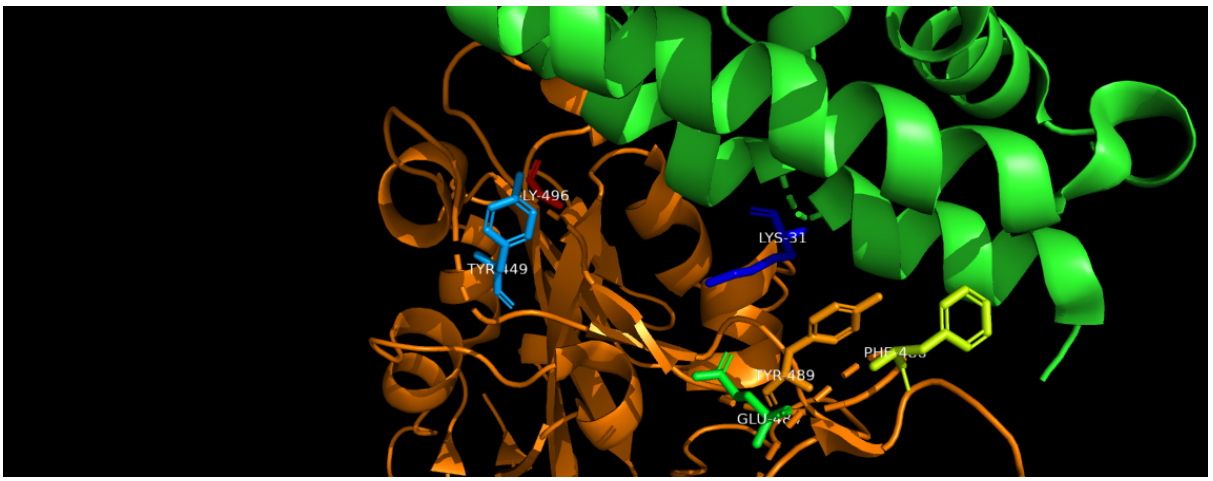
LYS A31, LYS A353, TYR E453, PHE E486, TYR E489, THR E500, ASN E501, TYR E505

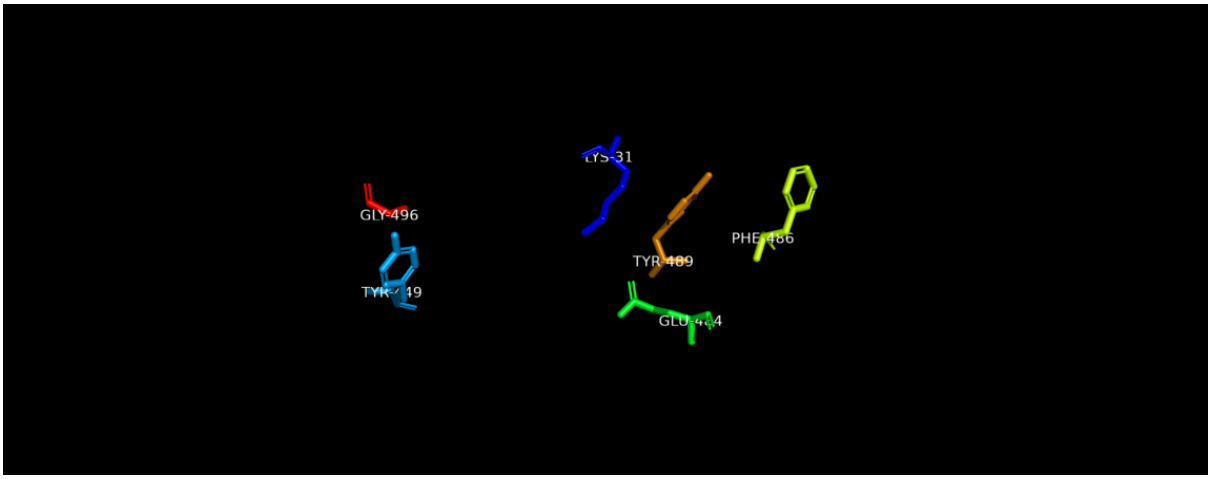




Solvation A-B

LYS A31, TYR E449, GLU E484, PHE E486, TYR E489, GLN E496





Result discussion:

After all the steps and computations we see that the residues that experience the most energy change when changes into alanine are the ones that do the interaction among the chains. The fact that they are the ones that experience the most energy change means that they are the most important for the complex, and it makes a lot of sense that the most important ones are the residues that conform the interaction among chains in a protein-protein interaction.
As a conclusion, we can say that the most important residues are those who are closest to the other chain, thus meaning that are the interaction point between chains and for that, they are the reason the interaction is taking place.

Issues encountered:

In this section we will explain the problems that we encountered along the way and what we did to solve them.

- NACCESS: When we tried to execute the naccess resource, we got a problem with permissions and the program did not execute. The issue was related to the path that we needed to change.
- Charge not found: When we executed step two we got an error related to the fact that the program did not find some of the charges of the atoms. We solved it manually in the library by modifying the names of the residues which had a suffix impeding the program from recognizing them. After looking at all the errors, there was still an error with the OXT molecule inside the Glycines. Since we could not find it anywhere to solve this we simply commented the line that printed the error so the error would be ignored.
- Running the python scripts, we did the code in a .py file to test it out, and when it was finished we would add it to the jupyter, as a python script is a more controlled environment. Some of the code we created would only work if launched through the terminal, so we had to modify it and adapt it so that it would work with jupyter, which took longer than expected.
- Another issue involving python code and jupyter was that a member of our group has a different python version than the rest, so when he worked in the jupyter notebook the code broke for the rest, this is another motive for which only the “clean” code was written on the jupyter.