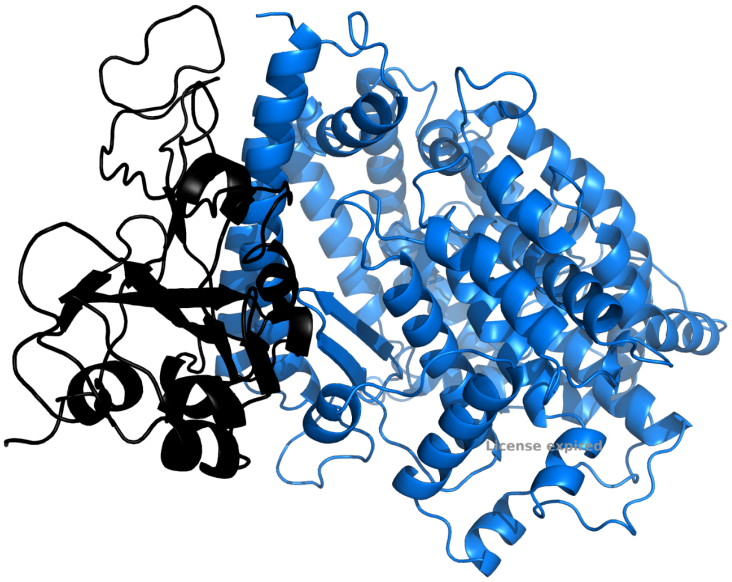
Protein-protein interface energy analysis

Biophysics



Adam Koershuis - Oriol Leal - Victor Mendez

22/11/2023

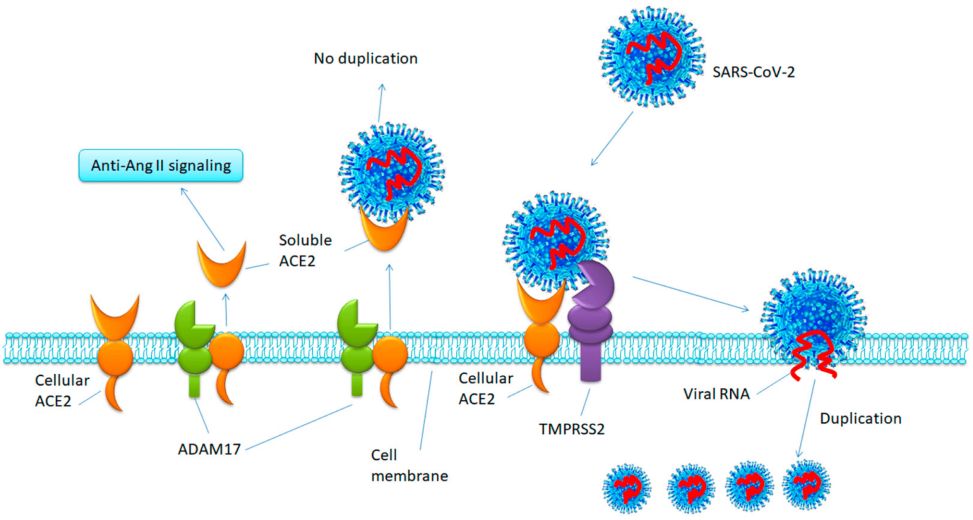
Table of contents

1. Scientific context

1.1. Spike and ACE2 proteins

Understanding the details of virus structure is crucial in combating infections. This analysis focuses on evaluating the interaction between the Receptor *Binding Domain* (RBD) of SARS-Cov-2 Spike protein and its receptor on target cells, the *Angiotensin Converting Enzyme* (ACE2) .

The S1 spike protein is a viral protein responsible for the initiation of the infection process. Within the protein, a specific region known as RBD is crucial for binding with high affinity to the ACE2 receptor on human cells. The ACE2 protein, found on the surface of various human cells, serves as the entry point for the S1 protein of the virus. Despite its role in viral infection, ACE2's primary function lies in the regulation of the renin-angiotensin system, a key mechanism involved in blood pressure regulation.



1.2. Objectives

The primary goal is to identify the interface residues essential for complex formation. By assessing the relative contribution of these residues through an Ala-Scanning experiment, we aim to comprehend the binding process and interaction energy before and after complex formation.

This knowledge could lead to the development of drugs or vaccines aimed at blocking this binding, effectively preventing viral infection. Additionally, the study will explore various SARS-CoV-2 Spike variants to broaden its scope.

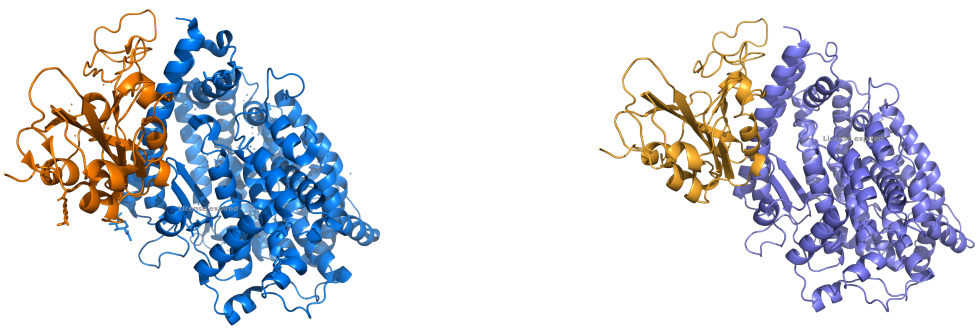
2. The data - preparation

To perform this analysis in the correct way, we first need to make a quality check on the provided data. In order to do this, we must consider that we want to study the protein complex in the most natural way possible, in other words, as we would find it in natural environments, since in the Protein Data Bank (PDB) most of the proteins are not entirely “natural”.

These preparation steps of our protein complex are essential for the subsequent energy analysis. When analysing a protein-protein interaction, it's crucial to focus uniquely on the protein components that make up the complex. Removing heteroatoms effectively trims away all non-protein elements, leaving behind only the amino acids that are part of the protein complex itself. This step helps in working with the most natural and biologically relevant representation of the protein complex, so there are no other elements that do not belong, naturally, in the molecule. This preparation step includes basic structure manipulations, protein backbone inspection or hydrogen atom addition amongst others.

We initially attempted to create our own code to perform the quality control check based on our Biopython seminar learnings *(remove_heteroatoms.py*). However, despite our efforts, our code did not work properly, which led to incorrect results. Therefore, we resorted to following the step-by-step guide from GitHub that was linked to the virtual campus (*preparation_steps.py*), even though we ended up writing part of the code ourselves. Our code iterates through the structure's residues, identifying non-hetero (main chain) and heteroatom residues based on their hetero-flag, and adds only the residues without heteroatoms to a new structure.

In the following images we can see a comparison between the before(raw) and after(cleaned) the preparation steps.



Left image shows the raw protein complex.Right image contains the cleaned version of the protein complex

3. Methodology and strategy

3.1. Determine amino acid residues that form the interface between the complex components

After doing all the preparation phase needed to filter the atoms in the structure of the protein 6m0j, we can finally work with the fixed model.
We are asked to inspect the protein visually using Pymol and for this purpose we followed a number of steps to identify the interchain residues.

- Open the 6m0j fixed structure in Pymol
- Use colours to distinguish between chains
- Using the function find look for interactions between chains
- Make a new selection of the residues that participate in the interaction (doing a selection of each chain helps in the process of selecting the interface)

We did the process 3 different times, one with the function find → polar contacts → between chains; find → any contacts → between chains within 3.0 A; find → any contacts → between chains within 4.0 A

Now, with visual help, we have to write a Python script that returns a list with the residues that are in the interface.

- The inputs are the structure of the protein, the ID of the chains we want to know the interface and the maximum distance between atoms we desire.
- The output of this script is 2 lists with the interface residues, one per chain.

In the script we will have to use some BioPython modules to work with the structure of the protein: PDBParser to parse along the structure and NeighborSearch to find atoms from another chain nearby an specific atom.

The structure of this algorithm is to iterate through all the atoms of one chain using PDBParser (Chain_1) and make a list of atoms from another chain (Chain_2) that are within a distance to those atoms. Once we have this list, we will add the residues whose atoms interact to sets, one per chain. With this algorithm if a residue from Chain_1 does not have any interaction, it will not be added to the set. However if it indeed interacts, the residue of Chain_1 will be added to the set 1, and all the residues participants in the interaction from Chain_2 will be added to set 2.

3.2. Evaluation of the complex interaction energy between the chains

After obtaining both lists for each chain, we are able to proceed. The aim of the second step is to obtain the interaction energy between components of a A-B complex with a python script or a notebook, in order to do this we will apply the following formula:

$$\Delta G^{A-B} = \Delta G_{elect}{}^{A-B} + \Delta G_{vdw}{}^{A-B} + \Delta G_{Solv}{}^{A-B} - \Delta G_{Solv}{}^{A} - \Delta G_{Solv}{}^{B}$$

This formula consists of the sum of the electrostatic interaction energy between components A and B, the van der Waals interaction energy between components A and B, the solvation energy between components A and B and the solvation energy for each component in isolation.

To start with our script we have defined a few functions that will help us to import the parameters of van der waals and the residues library (*get_params(self, resid, atid), def __init__(self, fname)* …).
Next, we have two functions: *residue_id(res)* and *atom_id(at)* that will generate readable IDs for residues and atoms, combining information like residue name, chain ID, and residue number.
'add_atom_parameters(st, res_lib, ff_params)' is the function of the script in charge of assigning atom type, charge, and van der Waals parameters to each atom based on the residue and atom information.

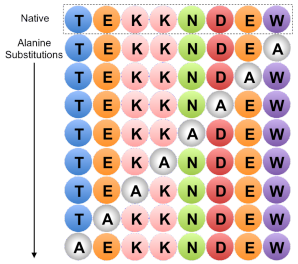
Then, we start to compute the Mehler-Solmajer dielectric function with the *'MH_diel(r)*' function. This function will help us to get the electrostatic interaction between the two atoms in the function '*elec_int(at1, at2, r)*'. After that, we compute the van der Waals interaction energy with the function *'vdw_int(at1, at2, r)*'. Also the solvation energy, based on the accessible surface area (ASA) of atoms in a residue with '*calc_solvation(st, res)*' function. And to finish with this part of the code we defined the function '*calc_int_energies(st, res)*' that will return us the electrostatic and van der Waals interaction energies for a given residue in the structure.

Now, we can jump to the final and most important part of the code, as it is the part that will give us the final results. This part is the one that we made combining almost all the functions that we explained in order to extract each of the variables needed to get the final interaction energy between chains 'A' and 'E'. There's a more detailed explanation of our code in the script.

Finally, with all the parts of the code needed, we are able to compute the interaction energy. We are going to compute (run the code) it using four different distances in order to know which is the best distance to obtain our desired result. The results and its interpretation are in the section 'Results'.

3.3. Ala-Scanning

Alanine scanning is a process that plays an important role in the understanding of the relative contribution of each individual amino acid in the protein complex, since it simplifies the chain and therefore gives us clues to the importance of certain residues. We use Alanine since it is a small, hydrophobic amino acid that can imitate properties of other residues, which can be used as a baseline for comparison.



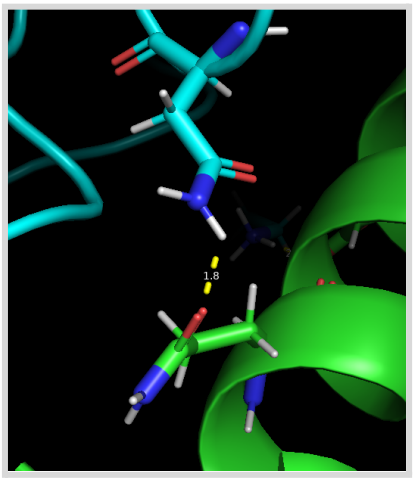
Example of alanine scanning. The native protein (top row) and each possible point mutation to alanine is considered.

For this step, using the list of interface residues, we replaced each amino acid with an *alanine* and then reevaluated the interaction energy of the protein-protein complex.
After executing the ala_escaning function in our script that performs the explanation above, we obtained the results in the following form:

- Column 1: chain ID
- Column 2: residue ID
- Column 3: position
- Column 4: solvation
- Column 5: solvation_ala
- Column 6: solvation_chain
- Column 7: solvation_ala
- Column 8: solvation_chain_ala
- Column 9: electric
- Column 10: electric_ala
- Column 11: vdw
- Column 12: vdw_ala

The last column, column 13, the absolute value of the interaction energies, was computed in R in order to make the plot in the results section, according to the formula shown in 3.2.

3.4. Pymol images



This is an example of interaction between chains found with Pymol's function to find polar interactions between chains. We can identify two residues, Glutamine in green from chain A and Asparagine in blue from chain E. In this interaction, a Hydrogen bond is created between Oxygen from Gln and one Hydrogen in the NH group from Asn.

4. Results

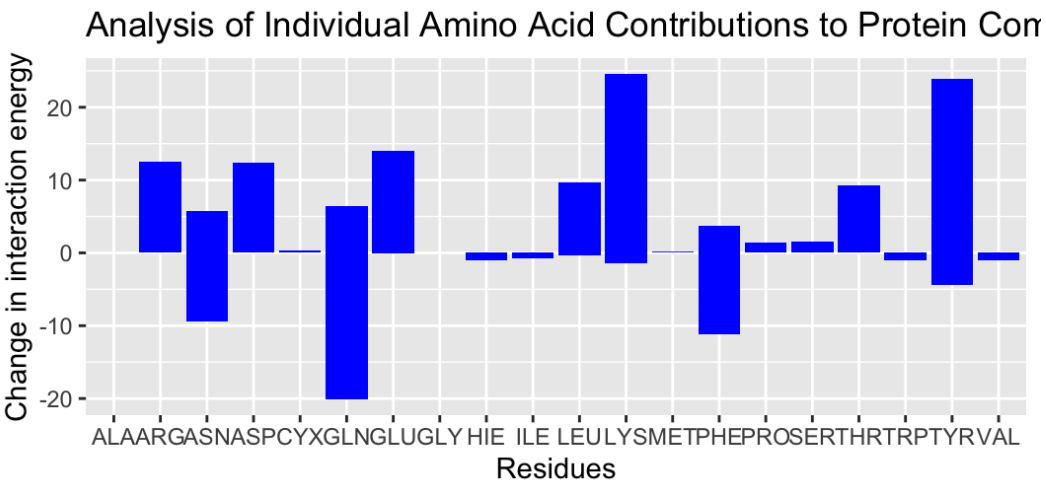
From the second step we obtained four results since we runned and, consequently, computed the interaction energy with four different distances:

Distance	Result
4	-120.80542817576911
5	-121.35695585544268
6	-130.76498559103968
7	-128.42934352684193

From these results we can extract two ideas: first, if we observe that all the values are negative we can conclude that there is an attractive interaction between the two chains (favourable). And in second and last place, in order to define which is the best distance to calculate the interaction in our case, we have to take a look at the results and see where the values reach a max. and then get stuck or decrease. In this case we see that this happens with distance 6 and 7, so this will be the best distances to calculate the interaction energy.

We discovered that a distance of 7 Å works well for spotting interface residues. This helped us create a list of these specific residues that can be found in the appendix.

After obtaining the list of interface residues and performing the ala_escaning function we made a plot to visualise the most relevant residues for the stability of the complex interface.



For this plot, in the Y axis we assess the value of the absolute change in interaction energy; in the X axis the different residues

- **Positive Values**: Indicates an increase in the interaction energy after substitution. This suggests that the original residue contributes favourably to the stability of the protein-protein interaction compared to alanine.
- **Negative Values**: Suggest a decrease in the interaction energy after substitution. This implies that the original residue contributes less favourably or unfavourably to the stability of the interaction compared to alanine.

The most relevant amino acids are Tyrosine, Phenylalanine, Lysine and GLN. As we can see, most of these mutations are negatively affecting the interaction energy. If we take a closer look at these amino acids, we can suggest that:

- **PHE** → has a bulky side chain which forms strong hydrophobic interactions within the protein interface. Since Ala has a smaller sidechain, it lacks this interactions
- **LYS** → has a positively charged side chain that can improve electrostatic interactions.
- **TYR** → has a similar sidechain to Phe, which is also capable of forming hydrophobic interactions
- **GLN** → has polar uncharged side chain that can form HDB, when replaced, it lacks this properties

5. Conclusion

The analysis of the SARS-CoV-2 Spike protein and ACE2 receptor interaction provided insights into the relevant residues forming the interface between the protein complex. Through Ala-Scanning, we identified specific residues like Tyrosine, Phenylalanine, Lysine, and glutamine, proving their significant roles in stabilising the protein-protein complex. Positive values in the interaction energy changes after substitution indicated favourable contributions of original residues, highlighting their importance in interaction stability, whereas negative values suggested a decrease in stability.

This detailed knowledge about how each amino acid helps in the connection is really important for creating new drugs or vaccines that can stop this binding and reduce how viruses infect us.

6. Bibliography

- *Bio.PDB.Entity module—Biopython 1.76 documentation*. (n.d.). Retrieved 22 November 2023, from https://biopython.org/docs/1.76/api/Bio.PDB.Entity.html
- *BioPhysics/Notebooks/6m0j_check.ipynb at master · jlgelpi/BioPhysics*. (n.d.). GitHub. Retrieved 22 November 2023, from https://github.com/jlgelpi/BioPhysics/blob/master/Notebooks/6m0j_check.ipynb
- Molecular Memory (Director). (2020). *PyMOL Tutorial: Modeling the SARS-CoV-2 RBD Interactions with ACE (COVID-19 Coronavirus Proteins)*. https://www.youtube.com/watch?v=hcnnKrlqa9M
- Xiao, L., Sakagami, H., & Miwa, N. (2020). ACE2: The key Molecule for Understanding the Pathophysiology of Severe and Critical Conditions of COVID-19: Demon or Angel? *Viruses*, *12*(5), Article 5. https://doi.org/10.3390/v12050491

7. Appendix

Interface residues in chain A: [<Residue PHE het= resseq=28 icode= >, <Residue LYS het= resseq=353 icode= >, <Residue LYS het= resseq=31 icode= >, <Residue TYR het= resseq=83 icode= >, <Residue ASN het= resseq=330 icode= >, <Residue HIS het= resseq=34 icode= >, <Residue GLN het= resseq=24 icode= >, <Residue ASP het= resseq=30 icode= >, <Residue ASP het= resseq=355 icode= >, <Residue THR het= resseq=27 icode= >, <Residue GLY het= resseq=354 icode= >, <Residue LEU het= resseq=79 icode= >, <Residue MET het= resseq=82 icode= >, <Residue GLN het= resseq=42 icode= >, <Residue ARG het= resseq=357 icode= >, <Residue TYR het= resseq=41 icode= >, <Residue GLU het= resseq=35 icode= >, <Residue ARG het= resseq=393 icode= >, <Residue ASP het= resseq=38 icode= >, <Residue GLU het= resseq=37 icode= >]

Interface residues in chain E: [<Residue GLN het= resseq=493 icode= >, <Residue TYR het= resseq=489 icode= >, <Residue ASN het= resseq=501 icode= >, <Residue GLY het= resseq=446 icode= >, <Residue GLY het= resseq=496 icode= >, <Residue THR het= resseq=500 icode= >, <Residue PHE het= resseq=486 icode= >, <Residue TYR het= resseq=453 icode= >, <Residue LYS het= resseq=417 icode= >, <Residue TYR het= resseq=449 icode= >, <Residue ALA het= resseq=475 icode= >, <Residue GLY het= resseq=502 icode= >, <Residue GLN het= resseq=498 icode= >, <Residue TYR het= resseq=505 icode= >, <Residue ASN het= resseq=487 icode= >, <Residue PHE het= resseq=456 icode= >, <Residue LEU het= resseq=455 icode= >]