Relevant links:
pytprot GitHub page: https://github.com/mariasr3/pytprot

pytprot PyPi page: https://pypi.org/project/pytprot/

1. pytprot: Theoretical background of the *pytprot* pipeline

Non-covalent protein interactions, such as hydrophobic effects, hydrogen bonds, and disulfide bonds are the functional basis of, if not all, most cells, as they allow for proteins to join into functionally diverse complexes. Therefore, the characterization of these interactions is essential to understand, on the one hand, how they drive many biological processes, and, on the other hand, how they actually form these macrocomplexes. Although there are several experimental techniques (X-ray crystallography, nuclear magnetic resonance (NMR) spectroscopy, and cryo-electron microscopy (Cryo-EM)) as well as computational approaches that shed light on visualizing and modeling protein complexes, these are limited. To overcome said limitations and to acquire more information new methods need to be developed [1].

The aim of this project is, therefore, to contribute to macrocomplex modelling based on PPI through a computational approach. To do so, we have developed a Python-based program, ***pytprot***, that reconstructs a protein complex by the assembly of several sets of interacting pairs. But, in order to fully understand the program's pipeline, we will need to explain some of the structural biology theory underneath it.

This program follows a specific pipeline in order to build the models. It is structured in three main modules (for more detail see section 3 of the GitHub *Tutorial*). The first step is the **Input recognition** where the input files, if correctly named, are preprocessed depending on whether the input is a list of interacting pairs or a macro complex. If the stoichiometry is provided, it is also parsed. Then, it has the **chain processing** module: in case that the input is a macro complex it obtains the interacting pairs, otherwise this step is not needed. After this, it searches for chains with sequence similarity. Finally, the last step is the **model construction** where the program uses the interacting pairs, the similar chains and the stoichiometry, if it is provided, to build the model using a recursive approach.

1.1. Chain interactions

Now, the basis of the model-building is the interaction between chains. Firstly, given the fact that we can have Protein-Protein chain interactions, and also Protein-DNA or Protein-RNA interactions, we have designed a specific function (see ***check_type*** in Section 4.) that recognises the two chain types. Secondly, we need to define what we consider an "interaction": Given the Cα or Cβ backbone of two chains, we will consider them to be interacting if said atoms are spatially close, within a distance smaller than 12 Å (literature indicates a range between 8-14 Å [2]).

The decision of computing the distance between just the Cα atoms, and not the full structure of the chain is because if not the computational cost would be really high.In addition, we will also consider the number of atoms that are spatially close, that is, the number of contacts. More specifically, we will consider two chains to be interacting if they have, at least, 8 contacts at 12 Å. Note that, in order to apply this to DNA chains, as they do not have a Cα, it is “obtained” through the conversion of their C1’, (see ***mutate_dna*** in Section 4).

Also, although using Cβ has shown better modelling results, Cα is still widely used as it is the backbone atom and some experimental methodologies, such as X-ray, only give information about this carbon [3], so this is the approach followed in our program. Both of these parameters are crucial for obtaining the interacting pairs from a macrocomplex input, as explained in the *Tutorial*, and therefore we included them as command-line arguments. Nevertheless, by default, the contact distance is set to 12 Å, and the number of contacts to 8. In addition, these have been implemented taking advantage of the **NeighborSearch** class from *Biopython* (see ***interacting_pairs*** function in Section 4.).

Nevertheless, this theory is only applied when dealing with a macrocomplex input, as what *pytprot* does is to split its chains, and search for the interacting ones with the ***interacting_pairs*** function mentioned above, which takes the contact distance and the number of contacts for doing so. At this point, *pytprot* also takes out the redundant pairs (see ***redundant_pairs*** function in Section 4.) to speed up the overall process.

An issue faced in this step is the handling of chain IDs when computing the interactions. In order to solve it, we have mapped every input chain ID to its corresponding number in the alphabet (see ***chain_processing*** function in Section 4.). We have included an “extended” alphabet, that contains both upper and lowercase characters, in order to allow for up to 52 different chains to be used as an input. The equivalence between the “old” and “new” IDs are also stored in a dictionary as we will need the original IDs when building the model.

Lastly, it is relevant to note that we have not deleted heteroatoms from those structures that had them. We just avoided them by filtering through their specific Bio.PDB atom.id (see the Biopython documentation). Nevertheless, this does not affect greatly the overall performance of the program.

1.2. Sequence and structural chain homology

Another crucial aspect of the *pytprot* macrocomplex assembly pipeline is based on high-homology chain pairing, both in sequence and structure terms. This will be a much-needed step prior to the superimposition, as we can only superimpose chain-pairs that contain a common chain. *pytprot* is based on a simple global pairwise alignment to look for similar-sequence chains (see ***similar_chains*** function in Section 4.).

For two chains to be considered as equal, we have given a normalized score (by dividing the alignment score by the length of the longest chain, in order to be more restrictive) to each pairwise alignment and established a cutoff of 0.95, which means that only chains with a 95% of identity or higher will be considered similar. At this point, it is to be noted that we assumed that the chains that were highly sequence-similar, would also be structurally similar. Although this may not be always the case, taking into account the structure conservation, the high sequence similarity, and the usual length of the macrocomplex chains (few of them are below 100 residues), we can be confident that the structures will be similar.

Also, protein structural modeling requires a stringent quality control to ensure that the final structures are *physically* accurate, otherwise, we could have artifacts such as steric clashes. These arise either due to low-resolution structures or homology models, and are characterized by the spatial overlapping of two non-bonding structures resulting in Van der Waals repulsion [3]. In *pytprot*, to detect steric clashes we have, again, resourced to Biopython'sNeighborSearch class, which searches if there are atoms closer to 2.5 Å between a pair of chains. This value is the minimal possible distance between two Van der Waals radii of two atoms interacting through a hydrogen bond (2.5 Å). This threshold will be employed in the model-building step, as the chains that will be added to the model will need not to "clash" with the chains present in the model. We only considered one clash per chain-to-be-added in order to not add it, which is a quite stringent threshold that may result in some final models not having some chains.

1.2. Superimposition

This will be the central aspect of the macrocomplex construction. With a set of chain-pairs that share common chains, after filtering for low-homology chains, we can build our model through interacting-chain-pair superimposition through the Superimposer module from Bio.PDB (implemented in the ***superimposer*** function in Section 4). Now, a central feature is the provided ***stoichiometry***, which will be used as a "guide" to build the model correctly. Nevertheless, it is an optional argument, so if not indicated, the model will add as many chains as possible, taking into account the steric clashes.

In any case, the model is built recursively, using a **random *seed*** upon which the model is built: To this random initial pair of chains, whenever possible, different chains from all the remaining chain-pairs with which they share a highly-sequence-similar chain will be added, as we assumed that in such case these chains are most likely to have a similar structure, and therefore, we can superimpose them.

For the superimposition, we will set the already-in-the-model chain-pair as a fixed model, and the to-be-incorporated chain-pair as a moving model. Then, after superimposing both pairs, we can obtain the rotation transformation or rotran matrix, and applying it to the moving model non-similar chain. But, before adding this chain to the final model, presence of possible steric clashes (only within Cα, as a simplification) is computed (see ***clash_list*** in Section 4). If no clashes are found, this new chain is added to the model. And then another chain-pair is compared with the structure, repeating this same process along all the different pairs of interacting chains.

It is relevant to note here that, as we might find situations in which the similar chains do not have the same number of residues, the alignment will be repeated and the program will consider only the shared residues between the two chains for superimposition.

2. *When* and *how* can *pytprot* be useful?

As we have already seen, the *interactome* is still yet to be studied thoroughly. In this process, having different types of macrocomplexes at hand can be useful in order to study specific PPIs. Within this context, computational methods such as *pytprot* could be used to build a specific macrocomplex given a specific set of experimentally determined interacting pair of chains and a given stoichiometry.

For the most general cases, that is, basic homo and hetero N-mers, *pytprot* can (almost always) accurately provide a correctly built macrocomplex. This can be optimal for studying the PPI between different chains of a given protein, or how the removal or mutation of a specific residue could affect the overall structure and its functioning. The consequence of the mutation, if it is in a critical position, will be that the interacting pairs found may be different and so such chain may not be added to the model or wrongly displaced. For a more clear visualization, we can use the macro-complex without the mutation and the new model mutated and visualize how the structure changes. Note that although mutations could also be employed to try to optimize the affinity between a macrocomplex and a given ligand (based on experimental and computational data???).

Also, some of the most important protein interactions are those that include nucleotides (DNA, RNA), either for gene expression regulation, DNA replication, translation or actually any biological process that involves handling DNA or RNA. As *pytprot* allows for reconstructing macro complexes with DNA chains, it can be used to study these DNA-associated chains with precision (see Section 3.).

2.1. Limitations

But *pytprot* also presents some limitations: Proteins for which the asymmetric unit consists of many different chains cannot be properly modelled with the stoichiometry provided. This can be a problem for higher-complexity multicomplexes, such as the ***6om3*** explained in the Github tutorial, or viral protein capsids, which are formed by the repetition of many multi-chain asymmetric subunits.

Also, really small chains are not usually added in the model, as these tend to not be processed correctly. Nevertheless, these really small chains are not usually implicated in the protein function, thus this should not present a big limitation.

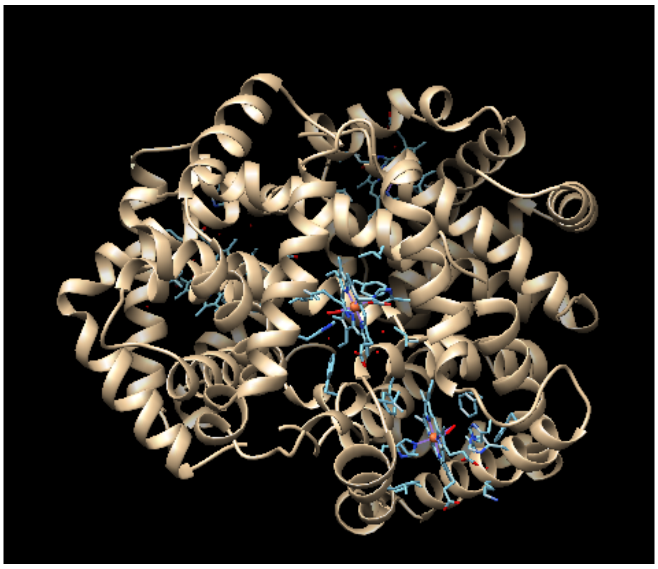
Lastly, another possible limitation of the model building approach is that it uses a random seed to generate the model. This results in the program only being able to output a single macrocomplex, which could be better or worse than any macrocomplex produced through a different seed, or a different run of the model. This needs to be taken into account when analyzing the results. It could be useful to further implement the model building through different seeds, and the comparison in structural or energetic terms of the built models, to select the best ones. Due to the lack of knowledge on the subject, this could not be properly implemented in *pytprot*.

In summary, with *pytprot* we expect to have good quality models when dealing with X-Ray-based models with 52 chains or less, that contain both protein and nucleotide chains, whose stoichiometry is known, and whose asymmetric unit is composed by only one chain. On the other hand, if we have biological assemblies conformed by multi-chain asymmetric units, *pytprot* can generate a model without stoichiometry, although it is likely to be badly modelled. Finally, for NMR-based models, or multi-assymetric unit assembly models, *pytprot* will not be able to produce any model.

3. Analysis of different models

For all the model figures, we will color the built models in white, and the reference structure in blue. Chains that are coloured differently will be indicated in their corresponding section. Furthermore, for all macrocomplex inputs, these have been built with a contact distance of 12 Å and 8 number of contacts, unless stated otherwise.

3.1. 1gzx



1gzx corresponds to oxy T state haemoglobin, a metalloprotein found in the red blood cells that works as an oxygen transporter. Its quaternary structure is the assembly of four globular protein subunits following an stoichiometry of 2 alpha and 2 beta chains (A2B2). This protein binds to two ligands, the heme group and the oxygen [4]. These interactions will be relevant for further analysis as they need to be the same in order for the function of the resulting macrocomplex to be conserved.

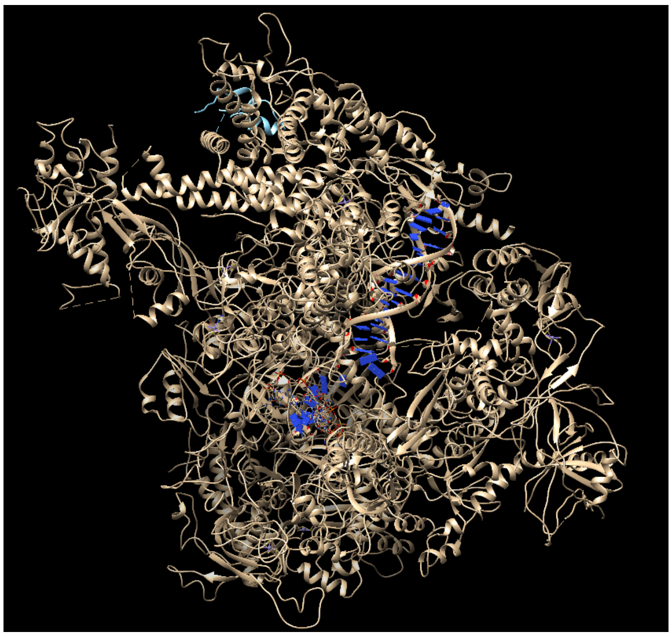
The reconstruction of this model is very successful, the resulting protein has the four chains well-located. We can see that the in superimposition that the reconstructed protein matches perfectly the original one with an RMSD between 146 pruned atom pairs of **0.000** Å; (across all 146 pairs: 0.000), as obtained from Chimera, which is not strange, as both structures match perfectly. The correct reconstruction of the model is also obtained if we do not provide the stoichiometry.

As everything is in place there is no reason to believe that the function is not conserved. Therefore all four chains may interact correctly with the heme group as well as the oxygen atom carrying out its transportation. Nevertheless, it is relevant to note that the final macrocomplex does not includes the ligands. If needed, these will have to be added afterwards.

3.2. 5fj8

5fj8 is the crystallized structure of the *S. cerevisiae* Pol III elongatig complex. It is formed by 17 different protein chains interacting with two DNA. The global stoichiometry is A1B1C1D1E1F1G1H1I1J1K1L1M1N1O1P1Q1. This protein is involved in transcription of DNA into RNA using the four ribonucleoside triphosphates as substrates [5].

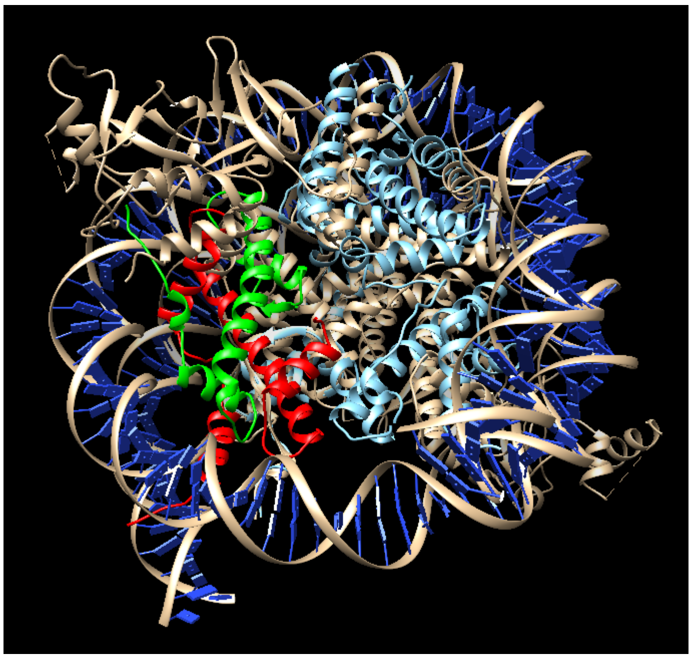
The reconstruction of the model using stoichiometry is very successful as can be seen in the following figure. The superimposition shows very good results with an RMSD between 1422 pruned atom pairs of **0.000** angstroms; (across all 1422 pairs: 0.000). Nevertheless, we see that chain Q is misplaced, although it is not clashing with any other chain. As this is not an essential chain (as we will explain later), this might not affect the overall function of the complex.



When the stoichiometry is not provided the resulting model lacks the Q chain. Note that the original pdb structure has bad resolution (3.9Å) that may affect the assembly process having a model with less chains. When providing the stoichiometry as we are "forcing" the chains that are not in the model we don't face that problem.

According to literature, chains that are involved in the main function of the protein are A, B and G as they participate in the formation of RPC1, RPC2 and RPC8 complexes necessary for DNA-dependant RNA transcription [5][6]. Chain B has some relevant residues that need to be conserved: on the one side C82-C34-C31 need a precise orientation and, on the other side, residues C53-C37 are involved in termination close to the non-template DNA strand [7] thus showing the importance to preserve the correct structure of chain B. In the resulting macro complex these chains are all perfectly placed when providing the stoichiometry. On the other hand, If said stoichiometry is not provided,

3.3. 6om3



6om3 is a crystal structure of the Orc1 BAH domain of *X. laevis* with a nucleosome core particle. This domain is part of the Origin Recognition Complex (ORC) and it is essential for replication, heterochromatin formation, telomere maintenance and genome stability in eukaryotes. It has an Hetero 10-mer structure with five different type of chains (A2B2C2D2E2) that interact with DNA. These chains are grouped forming histones H3, H4, H2A and H2B [8]. Note that the asymmetric unit is formed by two equal subunits having a total of 24 chains.

The final model results vary depending on whether the stoichiometry input file is provided. With stoichiometry the reconstruction of the biological assembly of macrocomplex is successful. However, when the stoichiometry is not given the chains C and D are missing.

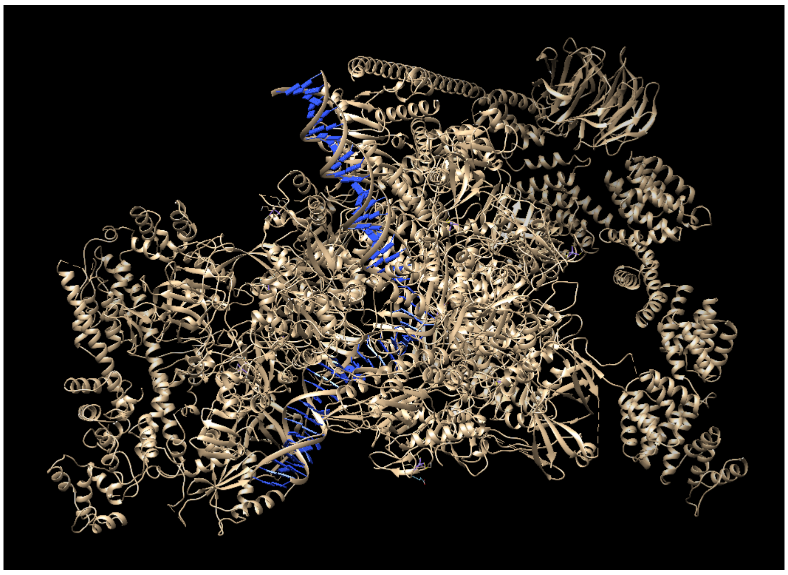
For further analysis the complete model has been chosen. In the following image we see that all chains, although they are not exactly the same as the original model, are in their place. The superimposition is quite good and the RMSD between 198 atom pairs is 0.000 Å, across all 198 pairs.

Residues 78-83 have been shown to be important for interaction of this domain with chromatin [9]. These are located in chain A that is perfectly superimposed to the original model (marked in red). Moreover, residues of the H4 tail recognition have been also reported to be important. H4 is formed by chains B, F and N [8] but in this case, the important residues (N60-A63) are located in chain B [9]. In the model we can see that chain B, colored in green, is also very well located. All these results altogether show that the function of the reconstructed model may be conserved.

3.4. 6gmh

6gmh is a crystallized molecule formed by 20 different protein chains (A1B1C1D1E1F1G1H1I1J1K1L1M1N1O1P1Q1R1S1T1) interacting with three DNA molecules. It is the structure of activated transcription complex Pol II, the factor b (P-TEFb), the elongation factors PAF1 complex (PAF) and SPT6 [10]. It is interesting to remark that this model has a resolution 3.10 Å which may influence the reconstruction of the model giving physically inaccurate results.

This protein has two ligands, one molecule of Zn and one of Mg, so the chains that bind them need to be correctly reconstructed in order to preserve the function of the protein. Chain A is interacting with both residues but Zn is also interacting with chains B, C, I, J, L.[10].



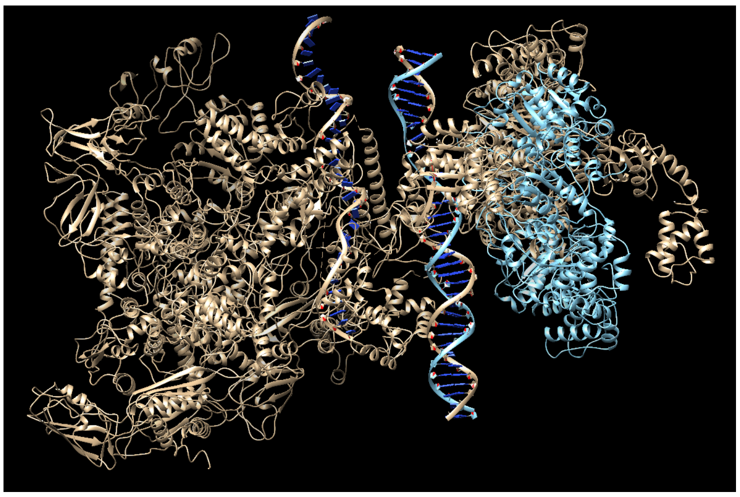
In addition, according to literature, the most relevant chains to preserve the function of the complex are B (DNA-directed RNA polymerase subunit beta), Y (Transcription elongation factor SPT4), Z (Transcription elongation factor SPT5), M (Transcription elongation factor SPT6) and W [10].

If we don’t use the stoichiometry the model obtained has 22 chains and with the stoichiometry the model has 20 chains. We see in the model that all the mentioned chains important to carry the function of this complex are well modeled. When superimposed to the original complex it matches perfectly with an RMSD of 0.000 A (across all 1411 atom pairs).

In this case the best model obtained is without considering the stoichiometry. This may be due to the fact that when using the stoichiometry the specific chains we need to add to complete the model have clashes with the chains that are already on it. This doesn’t happen without the stoichiometry as they are added if no clashes are found, which is a less stringent filter.

3.5. 5nss

5nss is a Cryo-EM structure of an RNA polymerase-sigma54 holoenzyme with promoter DNA and transcription activator PspF intermediate complex. This structure allows us to see how RNAp-o interacts with promoter DNA, which leads to the bubble formation. It also shows how the activor interacts with the RPc.

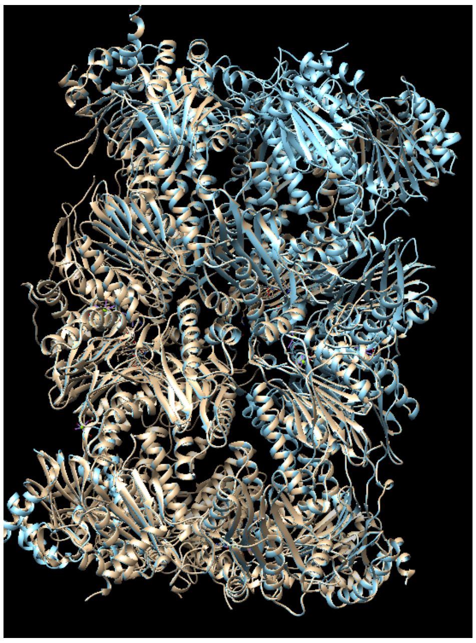


The reconstructed model has all 14 chains. Note that default parameters of contact distance and number of contacts have been slightly modified to 15 A and 3 contacts so the resulting model has all the chains, although misplaced. This is a good example to show how the **-d** (contact distance) and **-cn** (number of contacts) parameters should be modified. The default values have been chosen as a compromise between over-bloated complexes with different chains, and models with really stringent filters, that include a small number of chains. If these are increase, that is, if we want to be more flexible with the number of interacting chains obtained, this will likely result in the correct number of chains, but wrongly placed, as the rotran matrix applied to them before including them in the model is not the correct one, which is exactly what happened here. In such cases, we recommend sticking with the default values, where the final model does not have any big structural deformities.

Also, for this specific model, the chains that are not well located in the model are K, L and N, which are all from the operon transcriptional activator. This molecule inhibits the ATPase activity when interacting which allows the activation of psp transcription [11]. M is another important chain as it is the sigma-54 factor that makes the complex more stable as it recognizes promoter sequences. In this case the chain is well modelled and the interaction would take place [12].

3.6. 1g65

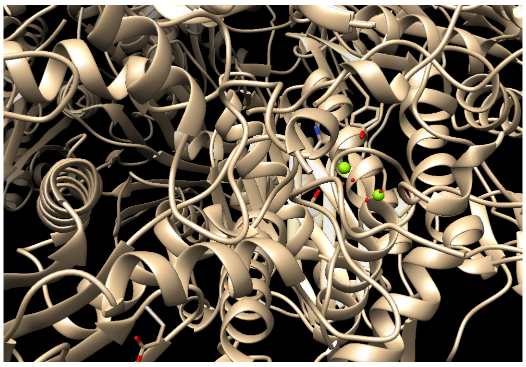
1g65 is the crystal structure of epoxomicin, an hydrolase that degrades poly-ubiquitinated proteins in the cytoplasm and in the nucleus. It is essential for the regulated turnover of proteins and for the removal of misfolded proteins [13]. It is an Hetero 28-mer (A2B2C2D2E2F2G2H2I2J2K2L2M2N2) with quite good resolution.



With or without stoichiometry we obtain the expected model with all 30 chains. Even though there are two of them (N and R) which are wrongly placed. Since the problem appears in both situations, it may be due to the seed given to construct the model. Another reason for the bad location of the chains can be due to the forcing step we used when the stoichiometry is provided as it “forces” the chains to be added but without applying any rotran matrix.

As this structure acts in a “compact” way, no chain can be missing or displaced without the protein function being disrupted, therefore this reconstructed macrocomplex will probably not have the functionality preserved [14].

Notice than we the model has also the ligand with which the protein interacts. This may be useful to see directly if our model has preserved its functionality.

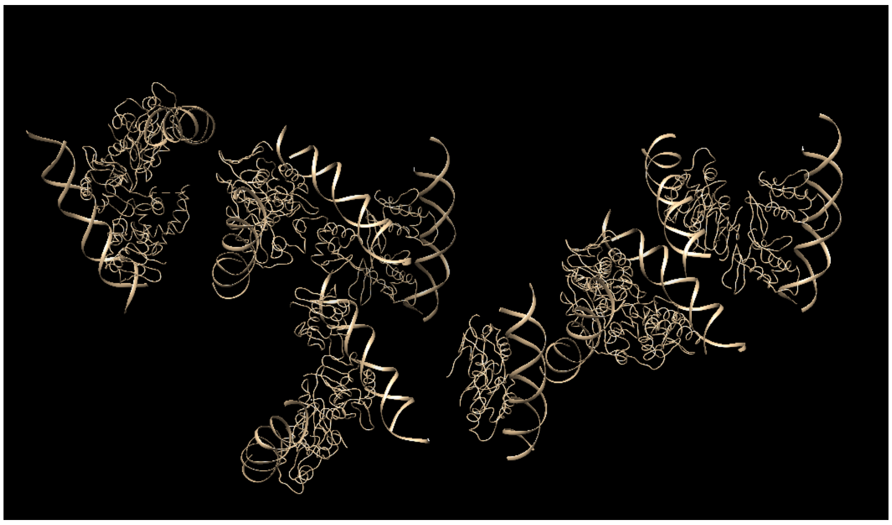


*Picture of the Mg atoms within the built **1g65** complex.*

3.7. 3t72

This last example is 3t72 or PhoB, a two-component response regulator that activates transcription when interacting with RNA polymerase.

The reconstruction of this model is very successful, all the chains are added and well-located. The superimposition with the original structure is perfect, as we can see in the following image, and therefore, the function is preserved.



4. In-depth description of the *pytprot* submodules

pytprot.py: It is the skeleton of the program. The other modules are imported here and it calls the specified functions to process the input files and create the model, which will be saved as a pdb file.

inputfunctions: module with which input chains are processed. Contains the following functions:

- **marcocomplex_parser:** Builds a dictionary containing as values a structure object, when the input given is a macro-complex (a single .pdb file).
- **pdb_parser:** Given a set of interacting pairs of chains as an input, it creates a dictionary with their names and Structures.
- **stoichiometry_parser:** Parses the stoichiometry.txt file in its correct format.
- **chain_processing:** Outputs the equivalence between old and new chain IDs and creates a specific dictionary type, also outputs the specific number of protein and nucleotide chains.
- **check_type:** It checks the type of the chain, which can be DNA or protein.
- **mutate_dna_chain:** It changes the C1' in DNA chains into a "CA", for posterior use in the superimposition.
- **acnucseq_from_pdb:** it obtains the nucleotide sequence from a nucleotide chain.

chainfunctions: This module is used to build the dictionaries holding the different relationships between the chains.

- **interacting_pairs:** When dealing with a macro-complex**,** it creates a dictionary of those chains that interact with each other, based on the interaction distance and the number of contacts between them.
- **redundant_pairs:** It speeds up the performance of the program as redundant interactions are eliminated.
- **similar_chain:** It builds a dictionary of those chains that in chain_align have a 95% or higher similarity
- **chain_align:** it makes pairwise alignments between the chains **unique_common_chains: It returns** dictionary of pairwise similar chains (in terms of sequence or interactions)
- **unicommon_completer:** It completes the unicommon dictionary if the stoichiometry is provided.
- **steichiometry_check_simple:**If the stoichiometry is given it checks if it is possible with the input chains we have.

modelfunctions: module used to build the model with the information obtained from the input files and stored in dictionaries.

- **superimpose:** It takes a pair of models and tries to superimpose them by taking the first input as the fixed chain and the second one as the moving chain. Then, a rotran matrix is computed and applied to the second chain.
- **Get_atom_list** and **common_chain_res** functions are used to compute the RMSD of the superimposed chains.

- **clash_list**: It checks if there are clashes with the other chains in the model to add or not the chain in it.
- **model_construction:** Central pytprot multicomplex protein assembly function. With the result of all the chain pre-processing, it builds the final model
- **save_model:** this function saves the reconstructed model in its correct format in the provided output directory.
- **stoich_count:** Compares the current stoichiometry and the input stoichiometry while the model is being built.

References

1. Lepvrier, E., Doigneaux, C., Moullintraffort, L., Nazabal, A. and Garnier, C. (2014). Optimized Protocol for Protein Macrocomplexes Stabilization Using the EDC, 1-Ethyl-3-(3-(dimethylamino)propyl)carbodiimide, Zero-Length Cross-Linker. *Analytical Chemistry*, 86(21), 10524-10530.

2. Bolser, D., Filippis, I., Stehr, H., Duarte, J. and Lappe, M. (2008). Residue contact-count potentials are as effective as residue-residue contact-type potentials for ranking protein decoys. *BMC Structural Biology*, 8(1), 53.

3. Ramachandran, S., Kota, P., Ding, F., & Dokholyan, N. V. (2011). Automated minimization of steric clashes in protein structures. *Proteins*, 79(1), 261–270.

4. Proteopedia.org. 2021. *1gzx - Proteopedia* [online] Available at: <https://proteopedia.org/wiki/index.php/1gzx> [Accessed 14 April 2021].

5. Proteopedia.org. 2021. *5fj8 - Proteopedia* [online] Available at: <https://proteopedia.org/wiki/index.php/5fj8> [Accessed 14 April 2021].

6. PDBsum (EBI-EMBL). 2021. *PDBsum entry: 5fj8*. [online] Available at: <http://www.ebi.ac.uk/thornton-srv/databases/cgi-bin/pdbsum/GetPage.pl?pdbcode=5fj8> [Accessed 14 April 2021].

7. Hoffmann, N. A., Jakobi, A. J., Moreno-Morcillo, M., Glatt, S., Kosinski, J., Hagen, W. J., Sachse, C., & Müller, C. W. (2015). Molecular structures of unbound and transcribing RNA polymerase III. *Nature*, *528*(7581), 231–236.

8. PDBsum (EBI-EMBL). 2021. PDBsum entry: 6om3. [online] Available at: <http://www.ebi.ac.uk/thornton-srv/databases/cgi-bin/pdbsum/GetPage.pl> [Accessed 14 April 2021].

9. De Ioannes, P., Leon, V.A., Kuang, Z. et al. (2019). Structure and function of the Orc1 BAH-nucleosome complex. *Nat Commun* ,10, 2894 .

10. Proteopedia.org. 2021. *6gmh - Proteopedia* [online] Available at: <https://proteopedia.org/wiki/index.php/6gmh> [Accessed 15 April 2021].

11. Uniprot.org. 2021. *pspF - Psp operon transcriptional activator - Escherichia coli (strain K12) - pspF gene & protein*. [online] Available at: <https://www.uniprot.org/uniprot/P37344> [Accessed 15 April 2021].

12. Glyde, R., Ye, F., Darbari, V. C., Zhang, N., Buck, M., & Zhang, X. (2017). Structures of RNA Polymerase Closed and Intermediate Complexes Reveal Mechanisms of DNA Opening and Transcription Initiation. Molecular cell, 67(1), 106–116.e4. https://doi.org/10.1016/j.molcel.2017.05.010

13. Proteopedia.org. 2021. *1g65 - Proteopedia* [online] Available at: <https://proteopedia.org/wiki/index.php/1g65> [Accessed 15 April 2021].

14. PDBsum (EBI-EMBL). 2021. PDBsum entry: 1g65. [online] Available at: <*http://www.ebi.ac.uk/thornton-srv/databases/cgi-bin/pdbsum/GetPage.pl*> [Accessed 14 April 2021].