stoichiometry# MACROBUILDER
by Aitor, Jorge and Lilian

Table of Content

1. Introduction

`project_SBI.py` is a python script built as final project for both Python and Structural Bioinformatics subject from the Msc in Bioinformatics for the Health Sciences. The aim of this program is to reconstruct macrocomplexes from `pdb` files for protein-protein or RNA/DNA-protein.

2. Tutorial

2.1 Installation

Just download the good sh*it from github!!!!!!!!! It's free real state!!!!!!!!

Usage from the Terminal

Mandatory Arguments

In- and Output files
When running the program from the command line, the user has to give some specifications for example on the in and output files.

```
'-i','--input'
'-o','--output'
```

The program only accepts **folders as input**. With the input command the user should provide the folder where the PDB files of binary interactions are stored. In order for the program to work correctly, it is recommended to have the input folder inside a folder where additional files (i.e. the stoichiometry file) are also stored. The input folder should have no subfolder since the program will crush in those situations.

Similarly, the **output parameter** should lead to a path where a folder containing the output files will be created. In cases were the user wants to create a folder with a name currently in use, please check the **force argument**.

Optional Arguments

stoichiometry File

```
'-s', '--stoichiometry'
```

The stoichiometry argument is optional. If the user knows the stoichiometry of the final complex, he can provide the **Path** where the file containing the stoichiometry is placed. Depending on the type of complex, the stoichiometry file should be in a certain format.
Protein-Protein Complex:

	Protein-Protein	**Protein-Nucleotide**
Format	[filename1]:3	[protein_complex1]:1
	[filename2]:2	[protein_complex2]:3
Example	1gzx_A_D:3	P05412:1
	1gzxA_C:2	P15336:3

```
'-f','--force'
```

Force allows the user to overwrite and existing folder that shares name with the one provided for the **output** argument. The default value is set to false, so it will not overwrite the conflicting folder and the program will just crush

Be aware!, the current folder will be deleted after the program is run.

Logging

```
'-v','--verbose'
```

The verbose argument allows the user to receive notifications in the command interface as the program develops. **By default, this argument is set to False**, so unless the user calls the argument no notification will be displayed.

Max Iterations

```
'-m','--max_iter'
```

The option max_iter specifies the number of maximal iterations to go over the input files to build the model so the program will not run indefinetely when infinte structures are built or if no new chains can be added to the model. The default value is set to 100, thus, when provided a high number of files the program might exit before the structure is finished.

3. Theory

3.1. Method and algorithm

The program takes a general approach shared for both kind of inputs (Protein-Protien & Nucleic Acid-Protein).

From the general approach, our algorithm would access the input file and look for files ending with `.pdb` extension, this means they need to be extracted from any compressed format. It is not necessary to remove any ther file but please, notice it will not be taken into account as input file. However, if the notation of files is not the appropiate, an error message will be send to the command line for further information.

Once the files are chosen, they are processed to extract the information for the sequences using the function `read_pdb_files` available in the script functions.py. This functions benefits from the `PDB_parser` module in the Biopython package. For each file the structure (containing the id and the file) is extracted and alpha carbons structures are obtained for every chain inside a single file. Besides, the heteroatoms are removed since they may not be meaningful for the final protein structure (for example, water).

Once the heteroatoms are removed and the alpha carbons structures are stored, the sequence for the latter is obtained. In order to avoid maching complexes out of ligands and cofactors, we set a threshold of 25 residues lenght below which we will not use the given protein for our study.

Once this is done, the program will differenciate between protein-protein and nucleic acids-protein input files. This is done by calling to another function developed called `alpha_carbon_retriever`. This function takes as input a given chain and processes it in order to determine wheter it is protein or nucleic acid. This is done by looking at the residues in the chains:

- If it has *CA*, then it is taken as protein,
- If it has *C4*, then it is taken as a DNA, RNA according to the notation ('DA','DT','DC','DG','DI' for DNA and 'A','T','C','G','I' for RNA).

According to this, chain type is selected and the alpha carbo

3.1.1. Protein-protein

3.1.2. Protein-Nucleic Acids

3.2. Biological Problems

Recent studies suggest that proteins may not work individually, but they will rather form a complex with other molecules in order to fullfill certain functions. A good examples of this model of interaction can be found in the ribosome or enzymes like the NADH dehydrogenase. While traditional experimental techniques such as x-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy have been crucial in characterising the structure of a great amount of proteins, they have found little success when dealing with macrocomplexes given the large size and structural flexibility these molecules present.

However, with the rapid development of computers and the decay in the computational hardware cost, fields such as computational biology have greatly bloomed, and may help us predict the structure of macrocomplexes *1* in silico *1* based on the huge amount of data recovered by traditional techniques. Our project takes a simple approach to this problem by employing superimpostion between highly similar chains as a basis. Superimposition is defined as the procedure by which which two molecule structures (two proteins, two DNA/RNA molecules, etc) are placed in space minimizing the distance between the backbone atoms of both structures. If we were to compare sequence alignment with estructural alignment, equivalent residues would be the ones filling the same position in a multiple alignment (according to a sequence similarity score), in structural alignment equivalent residues would be the the closest ones.

Once two chains that can be superimposed are identified, it is possible to calculate translation and roation matrices so the coordinate system of both structures are identical. By equating the coordinate system, we are able calculate how diffent the equivalent chains are. There are multiple measurements available to evaluate the structural alignment, but the most simple one, the Root-Mean-Square Deviations (RMSD), was the one employed in this project. RMSD is based on the average distance between two sets of atoms, usually the backbone atoms (&#945-carbons in the case of proteins and C4 carbons in DNA/RNA strands) of the superimposed molecules. By convention, bellow a value of 3 both structures will be considered the same.

4. Analysis

5. Limitations

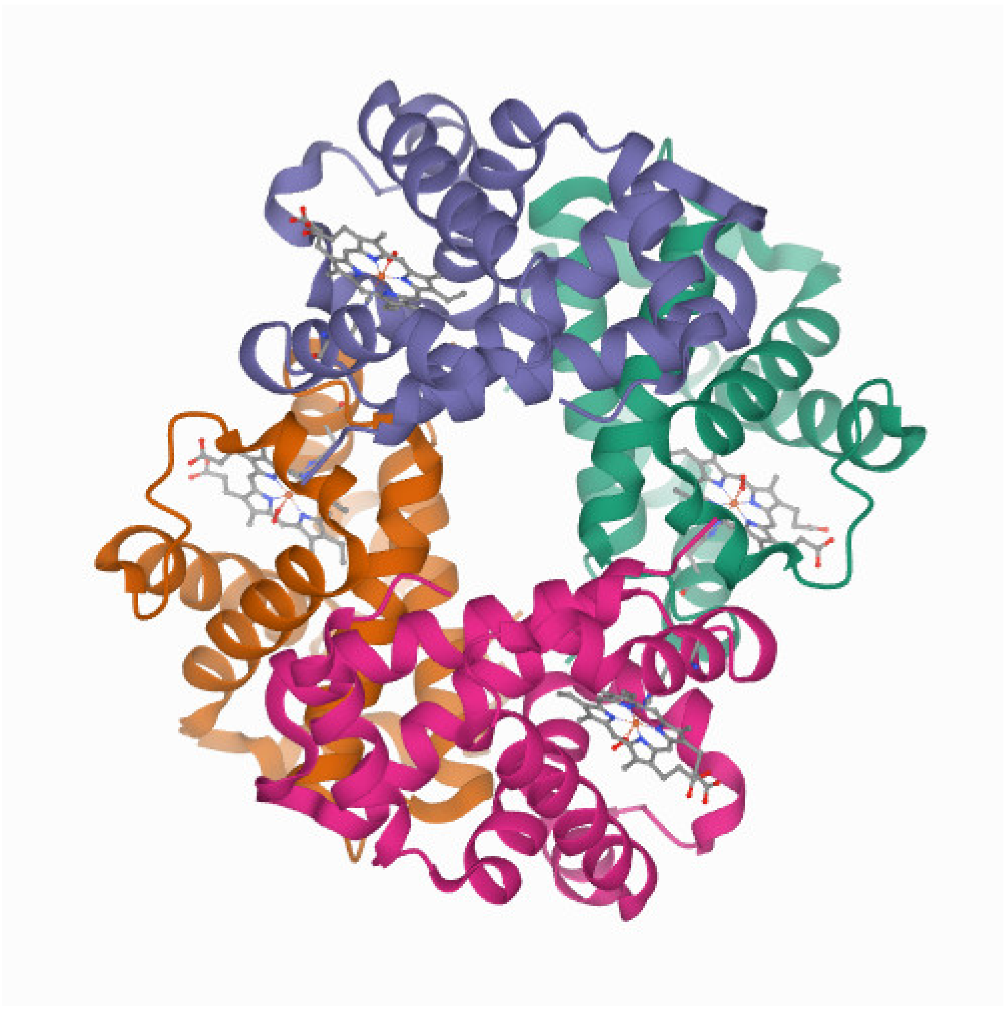
We note that our program comes with some limitations.

- for nucleotide protein we expect 1chain = protein, 2.+3.chain=DNA

6. Examples

6.1. Protein-Protein Complex: 1gzx

As the first simple example for a protein-protein macrocomplex, we will present the heteromer 1gzx. It is a heamoglobin consisting of four chains as it can be seen in the following figure.



The provided input folder consist of three different files:

- File1: 1gzx_A_B.pdb
- File2: 1gzx_A_C.pdb
- File3: 1gzx_A_D.pdb
  Other, than the name user might conclude from the file names, the chain A in File1 is not a homologue to the chain A in File3. Thus, the naming of the chains is not relevant for the builduing of the model. For instance, File2 contains the binary interaction of two homologue proteins.

Furthermore, we provide the user with a made-up stoichiometry file for this example so he can compare how the build structure might differ when including the *-s* argument. The stoichiometry file contains the following lines

```
A:2
B:4
C:1
```

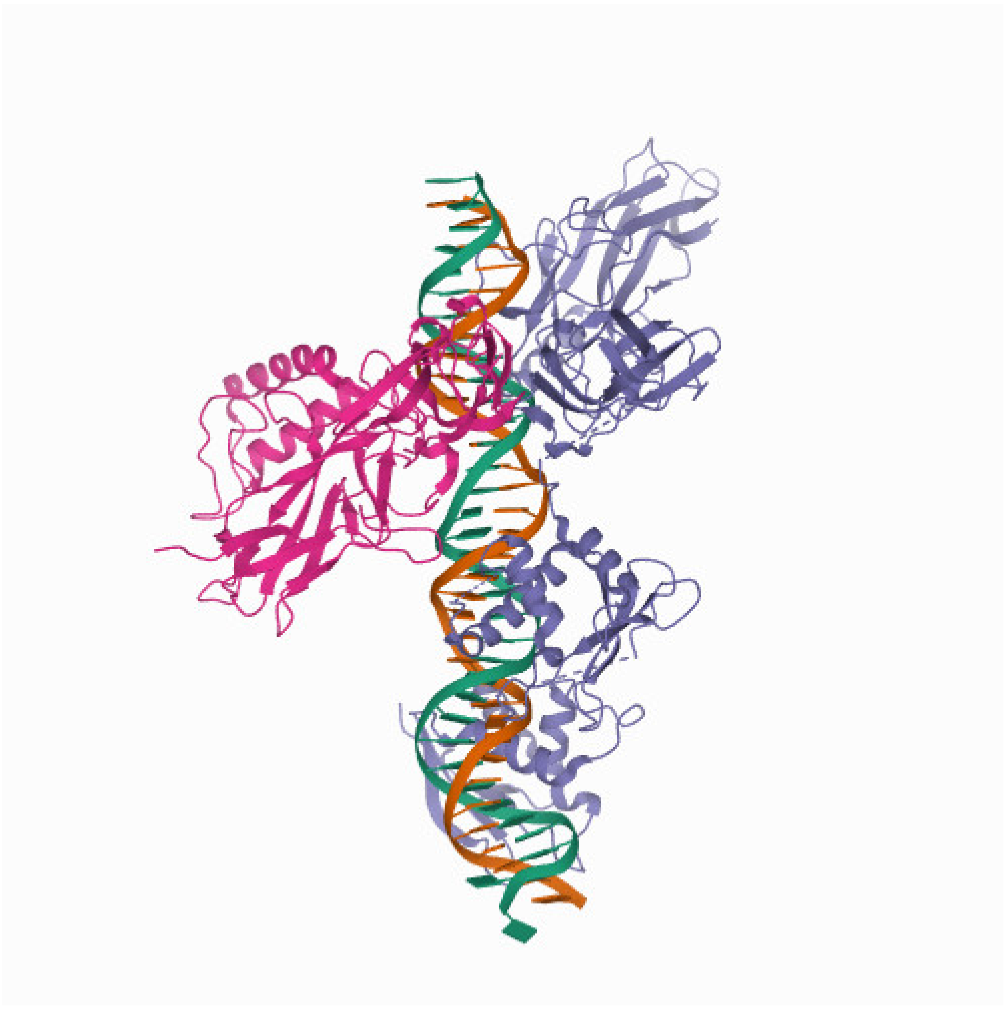
The final command has to be run on the terminal in order to build the macro complex of 1gzx:

```
python3 HERE GOES THE FINAL NAME OF OUR SCRIPT !"§$%&%$&/&%&/(&%/(/&%$§$%&/(/&%&/(/&%$%&/(/&%&/())))))"
```

The ourput is stored at **!"§$%&%$&/&%&/(&%/(/&%$§$%&/(/&%&/(**. If everything runs correctly, the model is expected to look something like the following:

6.2. Protein-Nucleotide Complex: 2O61

2O61 is an example for a protein-nucleotide complex. The model describes a structure of NFkB, IRF7, IRF3 bound to the interferon-b enhancer. Its 3D is presented below.



As this is a rather complex model, the input folder holds 54 pdb files containing binary interaction between one protein sequence and the according DNA strands. Thus, each file holds three chains. In contrast to the example of 1gzx, the naming of the files holds more information in this case:
{protein_name}.DNA.{pdb_name}*{chain}* {dnachains}.pdb

Example:
Q14653.DNA.2pi0_D_EF.pdb

This file holds part of the protein Q14653 which is Interferon regulatory factor 3 (IFR3, here chain D) in combination with a DNA double-strand (chain E and F).

As mentioned under the section describing the command option stoichiometry file, the user has to provide the stoichiometry in a different format than for the protein-protein complex. This is because for complexes including DNA, the stoichiometry file consists of the name of the sub complexes and the number of times they shall be included in the final model. For this example, the optional stoichiometry file provided contains the following lines:

```
P05412:1
P15336:1
P19838:1
Q04206:1
Q14653:4
```

When the user decides to provide a the path to the stoichiometry file in his command, the final complex will consist of one sub complex of each of the four first proteins and four times the sub complex Q14653.

The final command run on the terminal to conduct the macrocomplex of 2O61 is:

```
python3 HERE GOES THE FINAL NAME OF OUR SCRIPT !"§$%&%$&/&%&/(&%/(/&%$§$%&/(/&%&/(/&%$%&/(/&%&/())))))"
```

The ourput is stored at **!"§$%&%$&/&%&/(&%/(/&%$§$%&/(/&%&/(**. If everything runs correctly, the model is expected to look something like the following:

ENTER SCREENSHOTS OF OUR MODELS HERE AND FOR 1gzx