manual

Fabián Robledo

25 de marzo de 2020

Manual

Welcome to Promod's manual. Here you will find any information you need to run this software.

What is Promod?

Promod is python tool whose goal is to form macrocomplexes of molecules starting from the interacting pairs of chains that will form the complex.

Scientific Background

Tutorial

Installing

You can install Promod easily by downloading it from the github repository and running the next command in the downloaded folder. This is, for now, the recommended install way

```
pip3 install .
```

Alternatively, you can also run the setup script installation commands.

```
python3 setup.py install
```

Using

Promod has a command line interface which explains briefly how to use it before executing it.

```
python3 main.py -h
```

```
## usage: Build a macromolecular complex using interacting subcomponents
##        [-h] -i INPUT_FOLDER -o OUTPUT_FOLDER -f FASTA [-v] [-s STOICHIOMETRY]
##        [-d DISTANCE] [-t THRESHOLD] [-optimize] [-start START]
##
## optional arguments:
##   -h, --help            show this help message and exit
##   -i INPUT_FOLDER, --input-folder INPUT_FOLDER
##   -o OUTPUT_FOLDER, --output-folder OUTPUT_FOLDER
##   -f FASTA, --fasta FASTA
##   -v, --verbose
##   -s STOICHIOMETRY, --stoichiometry STOICHIOMETRY
##                        The stoichiometry of the final molecule the builder
##                        will try to forme
##   -d DISTANCE, --distance DISTANCE
##                        Maximun distance to consider that two atoms clash, in
##                        Armstrongs. By default, 0.1 armstrong
##   -t THRESHOLD, --threshold THRESHOLD
##                        Minimum score to consider two sequences homologus, and
##                        thus, the same to be build. Default 0.95
##   -optimize, --optimize
##                        Whether the model will be optimized with MODELLER
##                        after building or not. Default False
##   -start START, --start START
##                        Indicate the initial pdb from which the protein will
##                        be assembled
```

There are 3 mandatory commands while the rest are optional. The mandatory ones are related to the input files and output folder, necessary for the program to work, and they are *-i, the input folder; -o, the output folder; and -f, the fasta file of the sequences.* All of them will be covered in this manual.

Paramenters

Input folder The input folder should contain, at least, 2 pdb files. This folder may contain other files or subfolders; however, the program will ignore other files and will not check subfolders to find more pdb files. If you want to include some pdb, include it in the folder before running the program.

Beyond that last thing, no more things are necessary to know about the input folder. However, it's a mandatory argument, and should be indicated with -i or –input-folder tags

Output folder The output folder is the folder where the output files are written. This is mandatory, and no special folder is required. Just a warning, as, for the moment, the model file is written as 'final_model.pdb' and thus any other file with that filename will be overwritten.

The output folder is indicated with the -o or –output-folder tags

Fasta file The fasta file contains the sequences of the chains and the id for the stoichiometry. And it's a critical file, as the sequences must be homologous for the chain in the pdb. We can hold up to 5% of differences (by default, using -t parameter can modify this threshold). However, if one of the chains (if no stoichiometry indicated) or one of the chains in the indicated stoichiometry differs too much, the program will exit.

Note: That means the sequences in the fasta should be of a similar length than equivelent chain in the pdb. For example, the fasta file and the pdbs directly downloaded from Protein Data Bank may differ in the initial and ending residues, as they are quite difficult to model, and usually are removed from PDBs.

About the stoichiometry

Uninstalling

Examples

Example 1 (3e0d)

3e0d is a small complex formed by 2 protein chains and 2 double DNA strands. It's a subunit of the eubacterial DNA polimerase but it's perfect for an initial testing of our project, so we can test how the program works, the time it takes and if the result is similar to the original one. In this cases, we have 6 different interactions, as each DNA strand is counted as a single one interacting with another one and binded to the protein chain by one of them. In fact, this complex can be thought as a dimer: two monomers formed by a protein and a double strand DNA, which interact by some residues in their protein chains.

The original pdb downloaded from rscb looks like this:

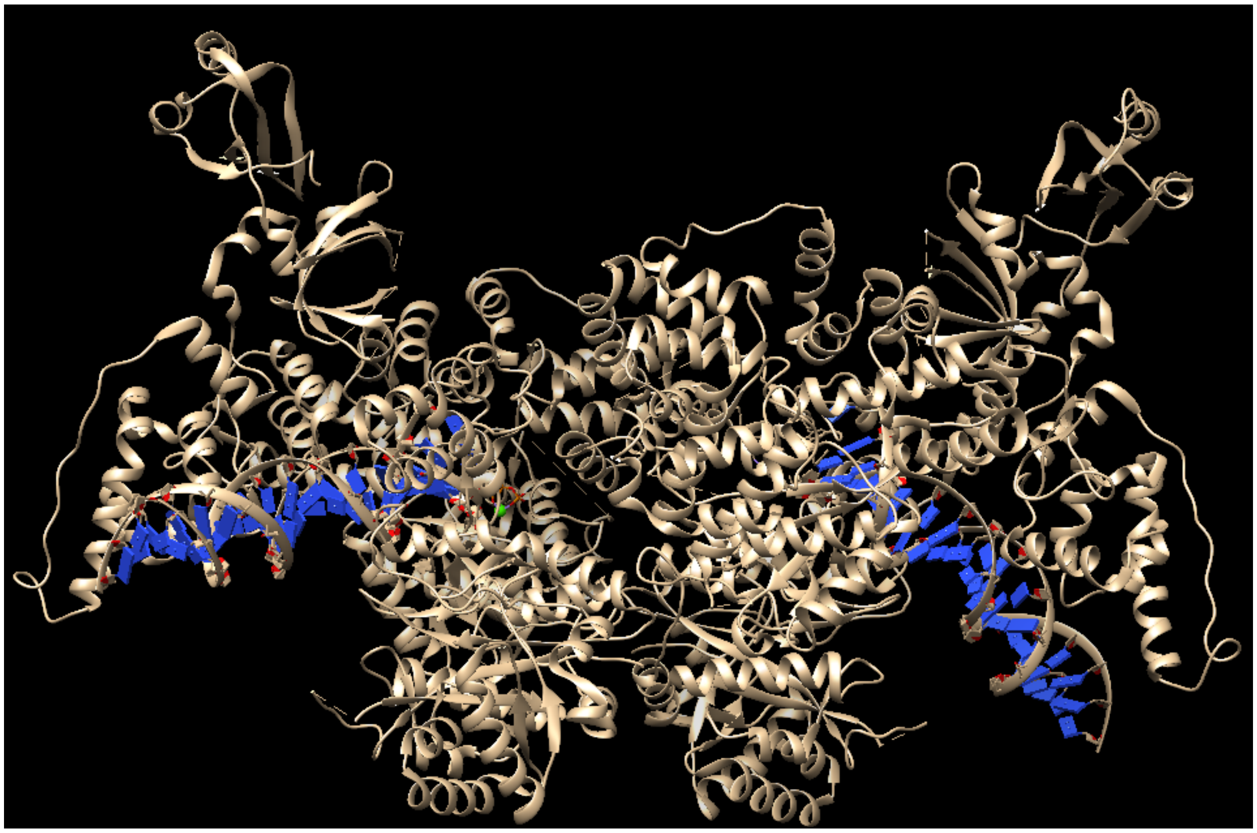

Figure 1: 3e0d structure obtained from rcsb.org

Non stoichiometry binding So, the first test to our program consists on joining the different pdb files into one single structure, without further information about the stoichiometry. Therefore, the program will try to build the protein with only the information provided by the pdb files and the fasta file. The command to build it is the next one:

```
main.py -i examples/example_1/chains/pairs -o examples/example_1/results  \
        -f examples/example_1/3e0d.fa
```

We just need to give the folder with the input pdbs (-i), the desired output folder (-o) and the fasta file with the sequences of the proteins. The program will take the different pdbs on the folder (By alphabetical order, to make it reproducible) And the result it's incomplete:

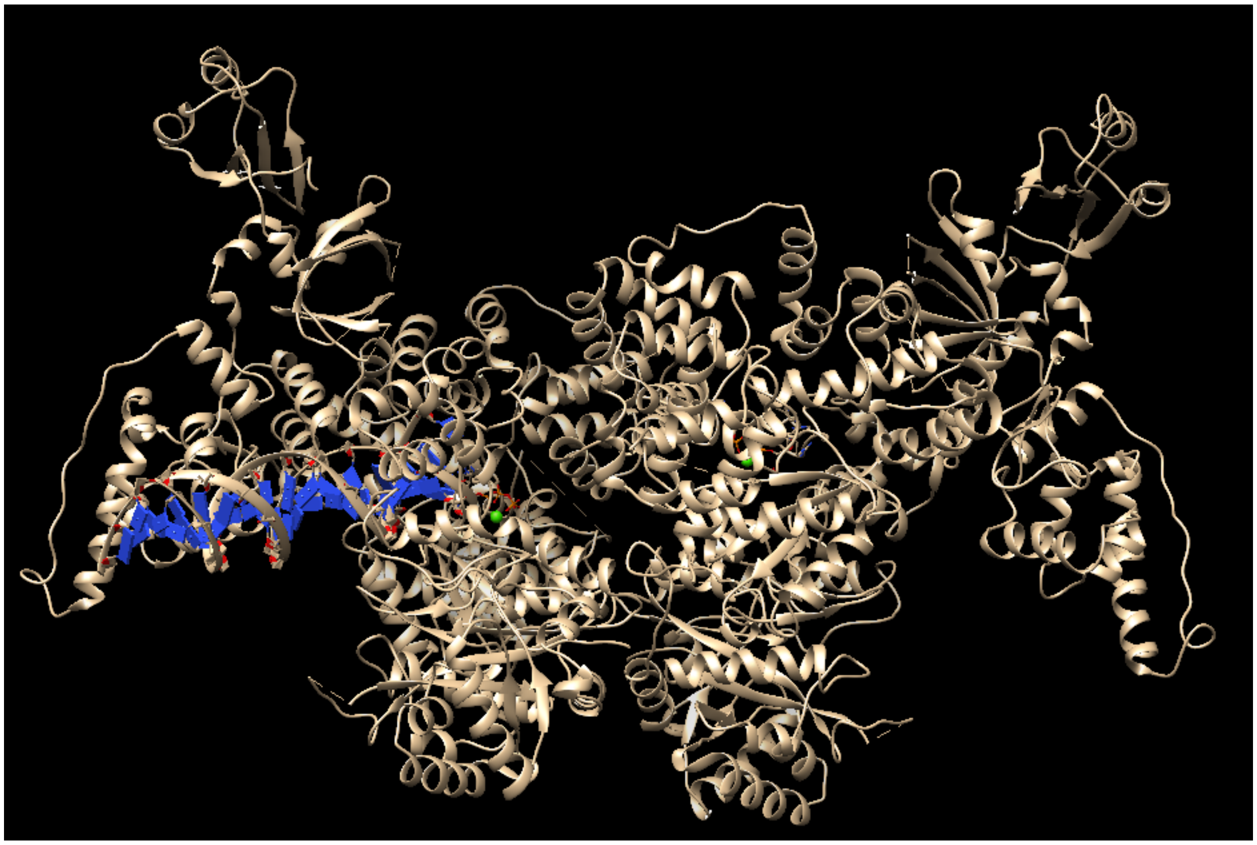

Figure 2: 3e0d builded structure from scratch without assistance

There we can see that there is a missing double strand DNA. So, let's see how we can complete the model. We can see that there is a missing protein, let's force the program to include it. We have two different ways of indicating this: The easiest one is selecting the starting complex of our model (which contains those missing parts). This results that different starting points can result in different models, so, it's important to select a good one. By default, the pdbs filenames are sortered alphabetically and the first one is chosen as the starting point.

The other not-so-difficult way is to indicate the stoichiometry of the complex. In this case we wil focus in selecting the starting pdb, so the command would be.

```
main.py -i examples/example_1/chains/pairs -o examples/example_1/results \
        -f examples/example_1/3e0d.fa -start examples/example_1/chains/pairs/3e0d_ZG.pdb
```

Yay! This is fare more similar to the original structure, very close to the original model. However, this is a very simple protein, and bigger complexes may are harder to build. However, from this example, we leart that:

- The program can build a model without any indication. However, it might be incomplete o have extra chains.
- The starting pdb is a important choice that can influence the final model. Thus it is important to remember which starting point gave which result. The same starting pdb will gave the same output.

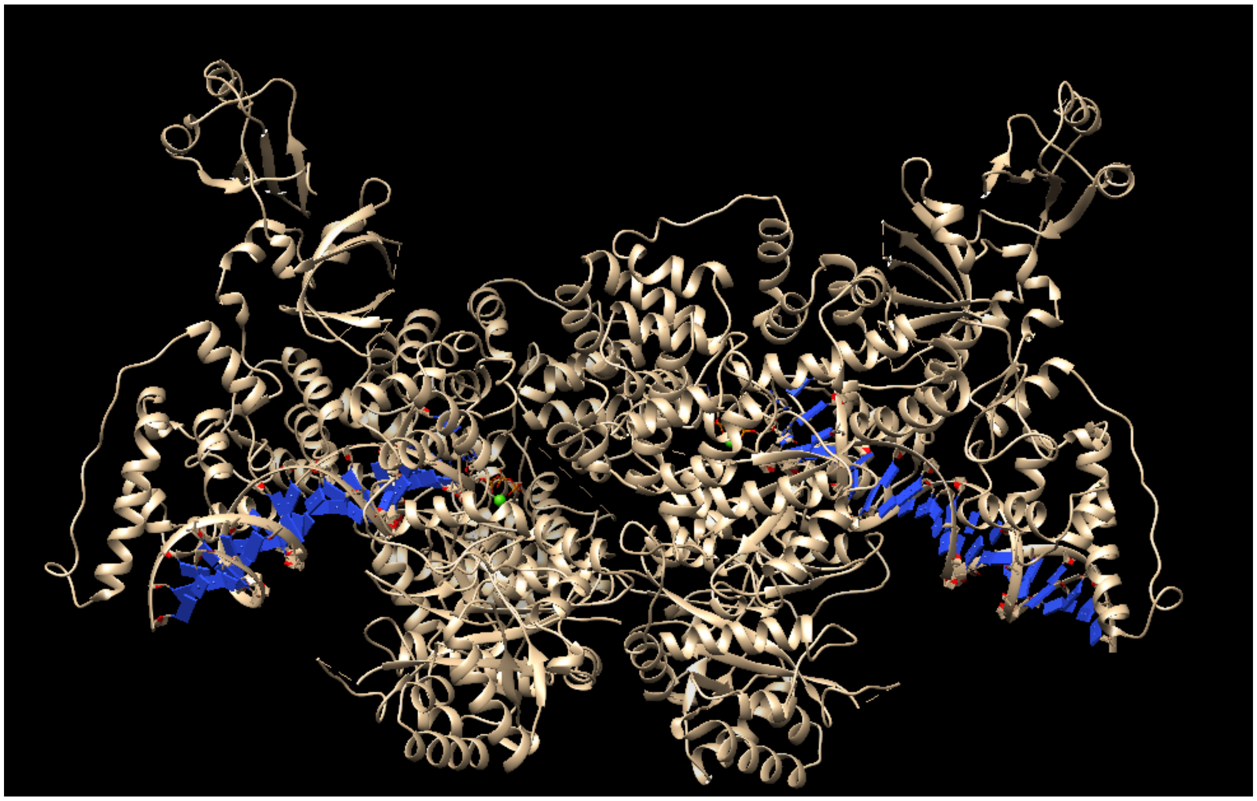

Figure 3: 3e0d indicating an starting pdb

Example 2

Example 3

Example 4

Example 5

Example 6

FAQ

Limitations

1. The sequences in the pdb and the sequences in the fasta must be similar. The program admits a certain degree of tolerance (which can be selected by the user using the -t parameter) but using sequences in the fasta which are fairly different from the pdb will not allow the assembly of the protein. This is particularly difficult for pdb in which protein tails are not well modelled and they're not present in the pdb but they're in the fasta. The pairwise alignment with the tail will drop the score of the alignment under the threshold, forcing to decrease it or even mistake to which sequence in the fasta corresponds to jus by chance. Thus, our recommendation is to prepare the fasta file with the sequence as similar as possible to avoid mistakes. In case you needed, in this package there is a script included, pdbsplit.py, which can give you the sequences of the chains inside the pdb file, avoiding this kind of errors up to ceratin degree. So, if you're looking for the effect of certain mutations in the desired protein, might be worthy prepare the fasta file according to the pdb sequences.

About Promod