ComplexArchitect - Tutorial

In this section we make a brief explanation of how to execute and use ComplexArchitect.

Command line arguments

The command line arguments that are needed to run ComplexArchitect are the following ones:

```
$ carchitect.py -h

usage: carchitect.py [-h] [-i INPUT] [-o OUTPUT] [-fa FASTA]
                     [-sto STOICHIOMETRY] [-m MODEL_NUM] [-C CHAINS]
                     [-clash CLASH_DIST] [-opt] [-v]

ComplexArchitect is a Python application designed to generate macrocomplex
structures from simple pair inetractions PDB files.


optional arguments:
  -h, --help          show this help message and exit
  -i INPUT            Input directory where PDB files with the pair
                      interactions are located.
  -o OUTPUT           Name of the output file, no extension is needed
  -fa FASTA           FASTA file with the sequences of the chains that will
                      conform the macrocomplex. They have to correspond to
                      the sequences of the chains from the PDB files. The
                      file should contain unique sequences; sequences don't
                      need to be repeated.
  -sto STOICHIOMETRY  Desired stoichiometry of the resulting complex. The
                      format should be like the follow example: A:2,B:4,C:1
  -m, MODEL_NUM       Maximum number of models the program is going to
                      build. The default number is 1.
  -C, CHAINS          Maximum number of chains that the resulting model
                      complex is going to have. The default number is 100.
  -clash CLASH_DIST   Minimum clash distance between 2 atoms. The default
                      minimum is 1.5 A.
  -opt OPTIMIZE       Refines the resulting model complexes and creates a
                      DOPE plot comparison between the unoptimized and the
                      optimized model.
  -v VERBOSE          Shows the progress of the program.
```

Input

This program needs an input directory with PDB files holding the protein pairwise interactions needed to reconstruct the desired macrocomplex. We can take into consideration that the same sequence appearing in different PDB files has not to be identical. it can handle 95% of identity. Each PDB file can consist on two interacting proteins, a protein and a DNA/RNA strand or 2 single DNA strands, dealing with those 3 scenarios.

Output

The output consists on 1 or more macrocomplex models built from the pairs of interactions provided. The output is stored in PDB format in a specific directory. If the optimized option is selected, the output will also include an optimized version of the model in PDB format and a DOPE comparison plot between the optimized and the unoptimized model. All output files will be stored in the same directory.

Mandatory parameters

The only mandatory arguments to run ComplexArchitect are *-i, input directory; and -o, output filename*. However, it is recommended to use also *-fa, fasta file*, since that way the user already identifies the chains with a given identifier and can specify the stoichiometry before starting to run the program.

Input directory

It is indicated with *-i* or *--input*. It consists on the name of directory with the PDB files of the interacting chains. The following considerations are taking into account:

- The directory name must exist.
- The directory must contain at least 1 PDB file.
- Each PDB file must contain 2 chains and no more than that. If a directory is not provided or any of the assumptions above are not accomplished, the program will reply and will finish its progression with a message indicating the specific problem. The program also ignores the existence of subdirectories or files with other formats.

Output name

It is indicated with *-o* or *--output*. The output name corresponds to the name that the different resulting complex models will adopt. If more than one complex model is created, the models will be named as *output_name*_1.pdb, *output_name*_2.pdb, *output_name*_3.pdb... The directory where all the output files will be stored also will adopt the output name in the same way *output_name*_results. It will be created if it does not already exists. As a warning, if the program runs with the specific output name several times, the resulting files will be overwritten.

Optional parameters

The optional arguments to run ComplexArchitect are the following ones:

Fasta file

It is indicated with *-fa* or *--fasta*. The FASTA file contains the sequences of the chains that will be used to construct the complex. Thus, the sequences in the FASTA file have to correspond or be very similar to the ones in the PDB files. The format of the FASTA identifiers that is allowed is based on the one that the PDB database uses. That format is the following:
>entry:chain|PDBID|CHAIN|SEQUENCE.
However, the program accepts a shortened version of that, since it ignores whatever there is after the specification of the chain.
>entry:chain
An example of identifier + sequence allowed by the program would be:

*>3t72:D
MDSATTESLRAATHDVLAGLTAREAKVLRMRFGIDMNTDYTLEEVGKQFDVTRERERKA\

The sequence do not have to be repeted more than once, as the FASTA file should consist on the unique chains with an alphanumeric identifier for each one. The FASTA file is recommended because knowing the identifiers for each sequence allows the user to add an stoichiometry before running the program, as an argument. That way, the stoichiometry should use the same identifier as the one in the FASTA file for an specific chain. The program uses the FASTA file to identify the chains that are equivalent among different PDB files, receiving the identifier from the file. A further consideration is that if the specified FASTA file name does not exist, the program replies with a message and finishes the process.

Stoichiometry

It is indicated with *-sto* or *--stoichiometry* when it is added as a command line argument. The stoichiometry allows the user to specify the number of times a specific chain has to be present in the resulting complex. The stoichiometry is provided as a string with the next format: *A:3,B:2,C:2,D:1.* The stoichiometry can always be added from the command line as an argument before starting to run the program. However, this makes sense when the user has already a way to identify each chain sequence with an identifier. This is the case in which the FASTA file is added. Thus, the identifiers from the FASTA file and the stoichiometry have to correspond to the same chain sequences. In the case in which a FASTA file is not provided, an alternative way for providing the stoichiometry is given. In that case, the program identifies the unique chains, analyzing which of them are equivalent. Then, the program adds a newly created identifier for each unique chain and all the equivalent chains will adopt that identifier. After that process, the unique chain sequences will be shown on the command line with the new identifiers. Then, since the user can identify the chains, the program offers to to add an stoichiometry with the same format as the one requested as command line argument. If a FASTA file is provided and no stoichiometry is specified as an argument, the programs assumes that the user does not want to add it and no request in the middle of the application is made. If the format of the stoichiometry is wrong, the program replies with a message and finishes the process.

Number of models

It is indicated with *-m* or *--models*. It corresponds to the maximum number of models that the program is going to build. Building more than one complex is encouraged, specially when dealing with unknown structures. The process of building the macrocomplex includes a random component that can vary the order in which the chains are added, resulting in different structures. This phenomenon is accentuated in cases with chains with many repetitions.

Maximum number of chains

It is indicated with *-C* or *--chains*. It corresponds to the maximum number of chains that the resulting models can accept. This can be an additional parameter to have better control over the resulting structure.

Distance threshold for clashes

It is indicated with *-clash* or *--clashes*. It corresponds to the threshold distance between the atoms from the candidate chain to be added and other parts from the macrocomplex, below which the program does not allow the addition of the chain. Below that distance, the programs considers a clash has occured. As default, it is set in 1.5 angstroms.

Optimization

The implementation is indicated with *-opt* or *--optimize*. If it is specified, after building the complexes and saving it in the specific directory, the structures are refined using MODELLER functions based on conjugate gradients. The optimized models are saved in the same directory with the name *output_name*_1_optimized.pdb. Additionally, a DOPE comparison plot between the unoptimized and the optimized model is created, saving an image file with the name *output_name1_EnergyProfile_plot.png. Furthermore, a file with the optimization stats of the complex (optimization_statsoutput_name*.txt) and files summarizing the DOPE profile of the models needed to make the plot (*output_name*_1_DOPE_EnergyProfile.txt and *output_name*_1_optimized_DOPE_EnergyProfile.txt) are also saved in the directory. The optimization process can take an extra time but it can be very useful to try reduce the energy of the resulting model, specially in cases in which the structure is unknown.

Showing progress of the program

It is indicated by *-v* or *--verbose*. If it is specified, the different processes that the program is performing are indicated in the command line.

Analysis of examples

To get a better understanding of how to run the program properly, we show different examples that represent different inputs that may be provided.

1jzx (hemoglobin)

Hemoglobin is a small complex formed by 2 homodimers. It is a very simple example to start with. In order to run the program we have to prepare the input PDB files in a directory. Being located in the 1jzx subdirectory from the examples directory, the command to build it is the next one:

```
carchitect -i 1jzx_pairs -o hemoglobin -fa 1jzx.fasta
```

The resulting model will be correctly built without specifying the stoichiometry. However, we can still include it without changing the outcome of the program:

```
carchitect -i 1jzx_pairs -o hemoglobin -fa 1jzx.fasta -sto A:2,B:2
```

Furthermore, we could get the exactly same result without providing the FASTA file. In the middle of the program the user could add or not the stoichiometry. Either way, the resulting complex would be the same.

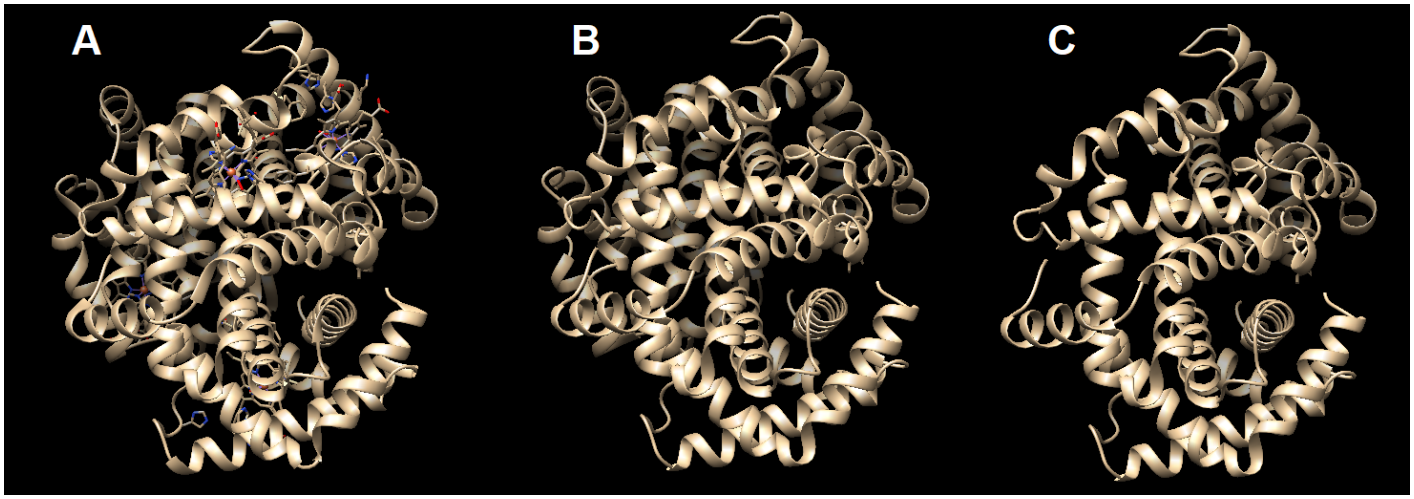
```
carchitect -i 1jzx_pairs -o hemoglobin
```

We could imagine a scenario in which we are interested in forcing to get a complex with a stoichiometry that does not correspond to the one of the native complex:

```
carchitect -i 1jzx_pairs -o hemoglobin -sto A:2,B:1
```

CArchitect also offers to limit the structure of the complex specifying the maximum number of chains to be added. This way, the model in construction will stop when achieving that maximum number of chains. In those cases, many different models can be obtained, since due to the random behavior of the application regarding the order in which the chains are being added, some models will have a set of chains that the others won't have. We could test to limit the number of chains of the hemoglobin with the next command:

```
carchitect -i 1jzx_pairs -o hemoglobin -C 3
```

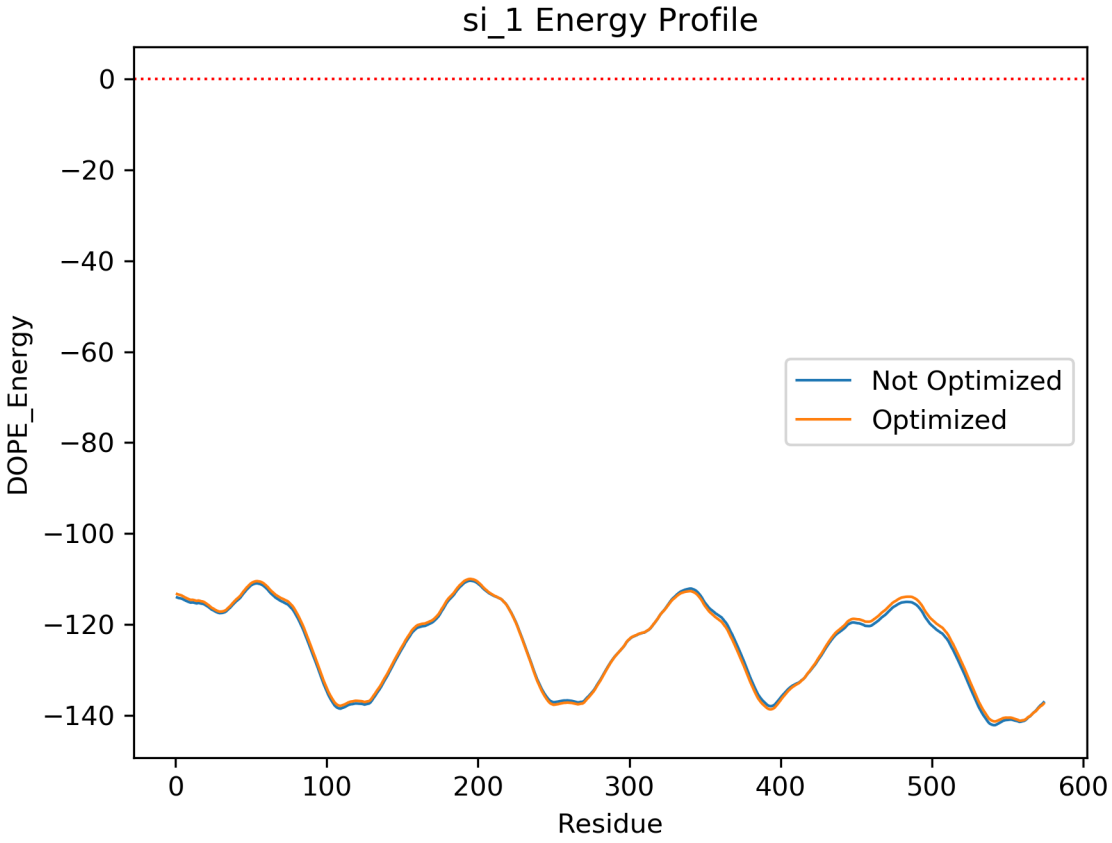


In Figure 1 we can see that the resulting complex model (B) obtained with the different equivalent commands is exactly equal to the native structure (A), except for the fact that small molecules (no proteins/DNA/RNA) are not included. Actually, the RMSD when superposing both structures is 0. We can also see the structure forced to have a non-native stoichiometry (A:2,B:1) (C).

```
ATENTION! You did not provide a FASTA file. However, you may want to specify the stoichiometry of
 the complex.
In order to correctly identify the sequences and introduce the stoichiometry, the sequences of th
e unique chains are provided below:

A:
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVN
FKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR

B:
VHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLH
VDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH

Do you want to specify the stoichiometry? [yes/no]
yes
Please enter the desired stoichimetry:A:2,B:2
```

In Figure 2 we can see an example of the response of the program if no FASTA file is provided. CArchitect ensures the user can identify each chain sequence in case stoichiometry specification is desired.
If we want to optimize the model, we can run the following command:

```
carchitect -i 1jzx_pairs -o hemoglobin -fa 1jzx.fasta -opt
```



As we can see in Figure 3, no big differences when comparing the energy of the unoptimized and the optimized model are found, but at least we have a plot that gives us valuable information about the energy profile of the model, being able to identify peaks and valleys that correspond to different parts of the structure with specific characteristics.

6gmh and 5fj8

6gmh and 5fj8 are the structure of the activated transcription complex Pol II-DSIF-PAF-SPT6 and the yeast RNA polymerase III elongation complex. They a very big complex formed by 20 different chains, with a double strand of DNA. In this case each monomer is unique; no repetitions of subunits are found. In order to build it, in the command line we can specify the stoichiometry or not. The resulting model will be exactly the same. Structures with simple stoichiometry can be built blindly. That is a strong point of the program.

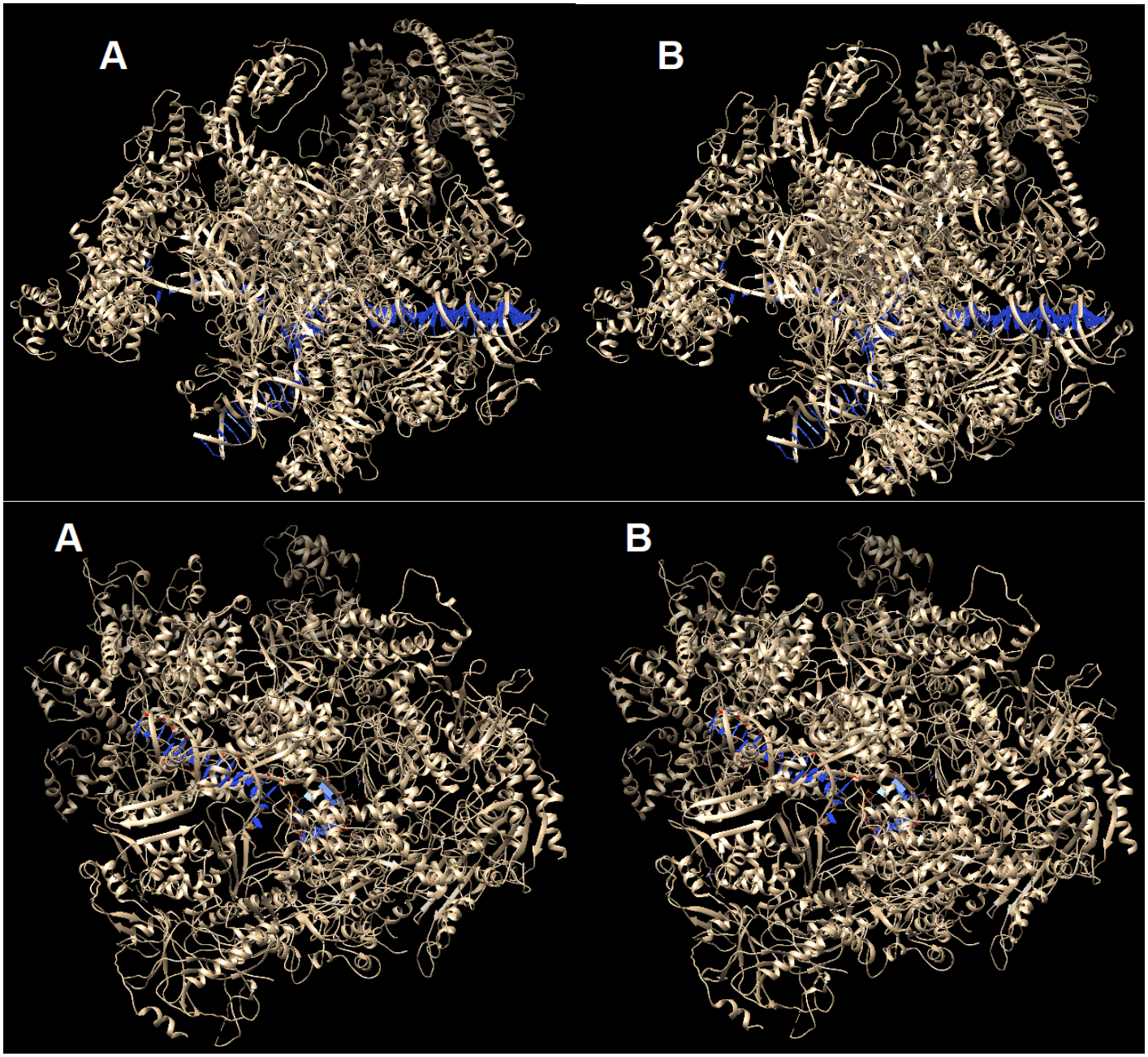
```
carchitect -i 6gmh_pairs -o 6gmh -fa 6gmh.fasta
```

```
carchitect -i 6gmh_pairs -o 6gmh -fa 6gmh.fasta \
-sto A:1,B:1,C:1,D:1,E:1,F:1,G:1,H:1,I:1,J:1,K:1,L:1,M:1,N:1,O:1,P:1,Q:1,R:1,S:1,T:1
```

\

```
carchitect -i 5fj8_pairs -o 5fj8 -fa 5fj8.fasta
```

```
carchitect -i 5fj8_pairs -o 5fj8 -fa 5fj8.fasta \
-sto A:1,B:1,C:1,D:1,E:1,F:1,G:1,H:1,I:1,J:1,K:1,L:1,M:1,N:1,O:1,P:1,Q:1,R:1,S:1,T:1
```



In Figure 4 and 5, when comparing the native structure (A) with the resulting complex model (B) we can see that they are totally equivalent. The RMSD of both structures when superimposing is 0. These models with a large number of chains take more time to be built. As the number of interactions grow, the time it takes for the program also increases. Since the process of the program is longer, we might be interested in visualizing what the program is doing in each step. We can do that adding the verbose parameter:

```
carchitect -i 6gmh_pairs -o 6gmh -fa 6gmh.fasta -v
```

3t72

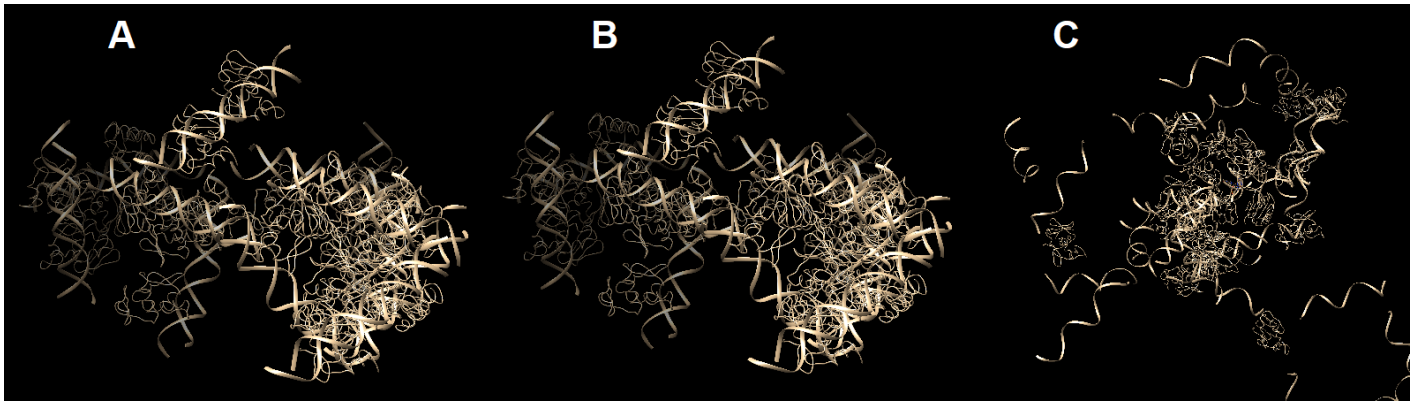
3t72 is a DNA Transcription Activation Sub-Complex. It is a good example because it is formed by many repetitions of the same monomers. Its stoichiometry is A:24,B:12,C:12,D:2,E:1 (although chain E could be an artifact). In those cases, if we do not provide the stoichiometry it is very possible that our model does not correspond to the native one. We can try that:

```
carchitect -i 3t72_pairs -o 3t72 -fa 3t72.fasta
```

Note that since we are providing the FASTA file, the program assumes we don't want to specify the stoichiometry.

Let's try it specifying the stoichiometry.

```
carchitect -i 6gmh_pairs -o 6gmh -fa 6gmh.fasta -sto A:24,B:12,C:12,D:2,E:1
```



In Figure 6, first we see the native structure of the complex (A), the resulting complex model specifying the stoichiometry (B) and the resulting model without providing the stoichiometry (C).
First, we can see that without specifying the stoichiometry, the model differs much from the native structure. This is due to the random behavior of the program. By chance, it may construct the correct model but it is not probable. Actually, the stoichiometry of that resulting model is A:20,B:22,C:12,D:4,E:1. On the other hand, when we provide the stoichiometry, the resulting model is very well constructed and the RMSD when superimposing with the native structure is 0. Thus, we always encourage to specify the stoichiometry to restrict the possible outcomes. We should advertise, though, that providing the correct stoichiometry does not always ensures that the correct model is obtained. Since the stoichiometry only restricts the maximum number of times the specified subunit will be present, it could be possible that that subunit does not reach the maximum. Fortunately, chances that that occurs are much lower when we specify the stoichiometry. Future perspectives would be also focused on fixing the minimum number of times a subunit should be present, not only the maximum. In the case that we are building a model blindly and we don't know its stoichiometry, we can always specify that we want to create several different models. We can always analyze its energy profile with the optimization function. We can do that with the next command:

```
carchitect -i 3t72_pairs -o 3t72 -fa 3t72.fasta -m 5 -opt
```

Finally, we want to stress out the importance of correctly identifying the chain sequences in order to specify the stoichiometry. That is the reason why we developed that alternative way to specify it.