Comparative Modelling of Elicitins with MODELLER

Organisms of the *genus Phytophthora* are parasites of crops and trees that are able to cause widespread devastation (*Phytophthora infestans*) is the causative agent of the Potato Blight). These organisms are morphological akin to Fungi, but belong in fact to a totally different Phylogenetic group, the Stramenopiles.

Elicitins are small extracellular proteins present in all *Phytophthora* organisms which are know to *elicit* a response in the infected plants. Although their mechanism is still unclear, it was shown that they can transport sterol molecules to and from the parasite and host organisms.

The 3D structure of two elictins, **cryptogein** (from *Phytophthora cryptogea*) and **cinnamomin** (from *Phytophthora cinnamomi*) has been solved experimental by X-Ray crystallography and NMR (nuclear magnetic resonance). However, there are hundreds of known elicitin sequences for *Phytophthora* and related genera.

In this tutorial we will use the MODELLER software to produce *comparative* structural models of two elicitins as of yet of unknown experimental structure. The first one (the easy case) is **capsicein** from *Phytophthora capsici*. The second one (harder case) is **oligandrin** from *Pythium oligandrum*. Being from a different genus, we expect oligandrin to be structural more diverse and as such harder to model.

Easy Modelling

We are going to build a model of capsicein (target) based on a *template* of known structure. We will need to look up the sequences on Uniprot (www.uniprot.org) and the structure on the Protein Databank (www.rcsb.org).

1. Go to Uniprot and and try to find the sequence of capsicein (Hint: use the organism name in the search, and the protein name "elicitin"). There should be only **one** protein named "capsicein" in the "Reviewed" section of Uniprot. Add it to the sequence basket.
2. We need to find out which, if any, structural homologues are available to model capsicein from. Go to the Uniprot page of capsicein, select BLAST (option "advanced") and in the following page select "...with 3D structure" as "Target database". This will search our sequence *against* all sequences of proteins with *known* structure.
3. Two sequences should stand-out, with a much higher sequence identity to capsicein (>80%) than the rest, cryptogein and cinnamomin. Further analysis of these structures on the RCSB Protein Databank (www.rcsb.org) will show that cinnamomin have a higher resolution. Compare the structures with and without a trapped ergosterol molecule. Select a template.

   NOTE: this would be a good time to run the "Clusters" Python script in the "Programmatic Access to the PDB" notebook. Use a cluster value of 100%, the PDB id of any structure of cryptogein or cinnamomin, and chain "A". Observe the results. Do the same with lower cluster threshold values. What can you conclude ?
4. Go to you Jupyter Hub session and open the folder "comparative_modelling" and then the folder "capsicein"

5. Browse the content of the various files by clicking on them. There are two *python scripts*, the PDB file of cinnamomin and three alignent files. This is all we need to run the MODELLER environment.
6. We are going to run the scripts directly from the *command line*, by opening a terminal window in Jupyter HUB. Once you do that, type

   ```
   cd comparative_modelling
   ```

   and then

   ```
   cd capsicein
   ```

   and you will be in the appropriate folder (check that you really are by running the `pwd` command). Then type the `ls` command to check the contents of the folder (the contents should exactly match what you previously saw with the Jupyter Hub file manager.
7. Thee following files should be present:

   capsicein.fasta - FASTA sequence file of capsicein

   capsi.ali - sequence file for capsicein with special MODELLER codes

   align2d.py - MODELLER script for structure-guided sequence alignment

   model-single.py - MODELLER script for building comparative models

   2a8f.pdb - PDB file for the template structure, cinnamomin.
8. Create the structure-based sequence alignment by running the command

   ```
   python align2d.py
   ```

   It should create two files, `capsi-2a8f.ali` and `capsi-2a8f.pap` The `.pap` file is good for visualizing the alignment, while the `.ali` file is the format needed by MODELLER for the next step
9. Create 40 models with the `model-single.py` MODELLER script, running the following command:

   ```
   python model-single.py
   ```

   After a short time, the script will stop and the folder should have a large number of files.
10. The 40 models are in PDB files named as "capsi.B999900xx.pdb" where xx are numbers running from 1 to 40.
11. Check the table at the end of the `model-single.py` output. There will be two numbers printed for each model, the pdf and DOPE scores. Better models will tend to have the lowest DOPE score.
12. The DOPE can be used to compare the quality of different models of the same protein, but it's not an absolute measure of model quality. To have that, we need to compute a *normalized* DOPE score, expressed in Z-score (number of standard deviations from the mean). Models with native-like quality will have scores around -1 or smaller, while positive scores are likely to indicate poor models. To compute a normalized DOPE score, we will use the `assess_model_DOPE.py` script, running it for all the models files in the folder, with the following command:

```
for a in *B9999* ; do python ~/Scripts/assess_model_DOPE.py $a ; done
```

(the above "for" command is a UNIX shell command that repeats the same actions for all files on a list - in this, the list of all files containing the `*B9999` in their name) Lokking at the output , you will see, that the normalized DOPE scores are all well below -1 .This is to be expected, because the high level of sequence similarity between target and template permits very good models.

13. Now we need to transfer the models to the local (classroom) computer, so they can be visualized with PyMOL and analyzed with other tools. To do this, first create a zip file:

    ```
    zip models.zip *B9999*.pdb
    ```

    (again, the expression "*B9999*.pdb" selects only the files we want) and then go the File Manager Interface in Jupyter Hub, click the check box left of the "models.zip" and select the "Download" option on the Menu Bar.

14. Extract the "models.zip" file to a folder in the classroom computer (or your personal laptop). In order to load, align and view the 40 models in PyMOL, we need another Python script (remember, PyMOL is a Python application and can run Python scripts), called "pymol_load_and_align.py", which you well download to your computer from the "Scripts" directory in your Jupyter Hub folder.

15. Open PyMOL and in the console type the following command:

    ```
    run pymol_load_and_align.py
    ```

    This will execute a python script inside PyMOL, and that script load all models and align them to the first model.

16. Now let's open the QMEAN site (https://swissmodel.expasy.org/qmean/) and load the model that had the best DOPE score on 12. Analyze the different outputs produced by the server. Run the same analysis for the experimental structure 2a8f.pdb and compare the results (QMEAN scores closer to zero are better.

17. On the Jupyter Hub server, there is a script called `get_qmean.py` that will call the QMEAN server and return the QMEAN4 z-score for a specific structure. To use it, simply tipe:

    ```
    python ~/Scripts/get_qmean.py <file_name>
    ```

    where `<file_name>` should be replaced by the name of the file you want to analyze (without the `<` and `>` characters). This scripts take considerable time to return the answer, often close to a minute.

Harder Modelling

We want to build an homology model for the beta-elicitin protein from the organism Pythium Oligandrum (a plant parasite), based on the known 3D-structure of another elicitin.

1. searching with protein name "elicitin" and organism name "*Pythium oligandrum*" will result in 7 hits, but none is explicitly called "oligandrin". Choose sequence C5NS42 and add it to the basket
2. Download the file in Fasta format (click on the protein code, select "Format", then "Fasta (canonical)"... Save the file. Keep in mind this sequence contain a signal peptide N-terminal region that will have to be removed (after alignment with the template).

3. From this point on, you should follow the same procedure as with capsicein, starting with searching for suitable template.
4. There is a folder called "oligandrin" inside the "comparative_modelling" folder.
5. In this folder there will be an `align2.py` and an `olig.ali` files, already prepared for you to run

   ```
   python align2d.py
   ```

6. Run 5. will produce the .ali and .pap files as previously. Now, to run modeller, there are three folders named `fast`, `slow` and `loops`. These folders contain each a different modeller script:

   model-single.py - the script we used to make the oligandrin models. It is simple, and runs very first, but may not be the best for target with a lower similarity to template or very accurate models.

   model-slow.py - this script has some parameter adjustments to produce a better refinement of the model, as well as more thorough optimization. With these setting the program runs slower, but models should be much better.

   model-slow-loops.py - this script is similar to model-slow.py, but with the added functionality of loop optimzataion: for each base model generated, a number of models with refined loops are created. In the present case, the script generates 10 base models and 4 loop optimizations for each, resulting in 40 loop-optimzed models (these are the modelos with names contain the "BL" letters.

   After generating the alignemnts with align2.py, copy the relevant required files (structure file and alignment file) to each folder to run modeller for the three scenarios.). To make the modeller scritps work, you need to fill in the blanks, accordin to the current file nanmes. To do that, you must edit the files using the a text editor. Here we will use the `nano` text editor. For instante, to edit `model-single.py` just use the command:

   ```
   nano model-single.py
   ```

   after making your changes, press Ctr-X and the Y to exit the nano editor.

   Compare the DOPE scores of the three sets. Copy the loop optimed models to your personal computer and view them in PyMOL, comparing with the template structure.
7. After properly editing the files, run modeller as before for each folder.Compare the DOPE scores of the three sets. Copy the loop optimed models to your personal computer and view them in PyMOL, comparing with the template structure.