01-Introduction to Bioinformatics.

November 5, 2022

[1]:
```
## This code block imports python modules to work with during the lab, you need␣
 ↪to run it first to get started.
# modules
import matplotlib.pyplot as plt
import nglview
import os
import shutil
from Bio.PDB import PDBParser, PDBIO, Select,  PDBList, MMCIFParser,␣
 ↪StructureAlignment
import Bio.Align
import os
from pathlib import Path
import rdkit

# local scripts
from scripts import viewer
from scripts import bio_align
```

```
/opt/conda/lib/python3.9/site-packages/Bio/SubsMat/__init__.py:126:
BiopythonDeprecationWarning: Bio.SubsMat has been deprecated, and we intend to
remove it in a future release of Biopython. As an alternative, please consider
using Bio.Align.substitution_matrices as a replacement, and contact the
Biopython developers if you still need the Bio.SubsMat module.
  warnings.warn(
```

1 Assignments:

The goal of this course is to design a new ligand for the target that was assigned to you. You will work in pairs. This ligand should dock with a higher docking score than the co-crystalized ligand. You will reach this goal by executing a bio-informatics part, a chem-informatics part, and finally perform docking.

Measurement Report: The results from your practical work are to be written down in a measurement report, which is to be handed in on Monday. This report should contain answers to the questions from the manual, and descriptions of what you did during the course (your train of thought). Naturally you should also include name, student number, etc.

Presentation: In addition, you are to hold a presentation on these subjects on Friday the 7th of October (afternoon). The presentation should last ~10 minutes with some subsequent discussion. In the presentation you should provide a background of your drug target and drug (type, pathology etc). There should be some results from the QSAR modeling and several docked poses. You should present your newly designed drug and defend why it is better.

The practical consists of three parts, which each have their own Jupyter notebook with instructions and the code to run:

- Intro to cheminformatics
- Machine learning and QSAR
- Working with crystal structures

2 Bioinformatics

2.0.1 Analyzing your Drug target

It is important to get a good feeling for the target you are about to study. This is where literature research comes into play. A good starting point is the original publication that your PDB code refers to. This will also give you further references. In addition, general reviews (Pubmed, Google Scholar) can be very helpful.

1. Describe the target (your report should contain at least the following information):

- Pathology (why are we interested in the target?)
- Clinical drugs available (which drugs are on the market) Also use ATC information to further describe the drugs.
- Protein information (what family of proteins is the target a member off, where is it located in the body?)

2.0.2 Related proteins (off-targets), based on sequence

Using Bio-informatics we are going to compare the target to other proteins. We use the primary structure (sequence) of the protein for this. Go to: http://www.uniprot.org/

2. You can find the link to Uniprot on the sequence tab on the page of your target on https://www.rcsb.org/

3. Alternatively, find the name of the target in the 'Protein Knowledgebase' (UniprotKB). From the list of hits you should select the correct one.

4. Verify the 'Accession' number with your supervisor. Yellow stars means that it is a correct and reviewed target, use a reviewed one unless you are working on H5N1.

In the case of the HIV targets, you have to make a sub selection within the POL polyprotein, do this by selecting the relevant domain in the list of domains below on the page. RT -> select the 'Reverse Transcriptase domain', 'Reverse Transcriptase Ribonuclease H' domain. Protease -> select the 'Protease domain'

5. When you have found the correct one, copy the following information:
    - Sequence Length / Status / Protein Existence
    - Mass (kDa)

6. Use BLAST (Basic Local Alignment Search Tool, under 'Advanced') to find identical proteins in other species (related target 1).

- Database 'UniProtKB', Threshold at 10, Matrix on auto, Filtering on none, Gapped on yes and Hits on 1000. Copy the following information.
  - Length
  - Identity
  - Species

7. Click on the 'Accession' number of the same hit, write the 'Accession' number down and copy the following information:

- Status / Protein Existence
- Mass (kDa)

8. Use BLAST to find similar targets in the same species (related target 2).

- Database '…Human', Threshold at 1, Matrix on auto, Filtering on none, Gapped on yes and Hits on 1000. Copy the following information of the first hit that is not an isoform of your target.
  - Length
  - Identity

9. Click on the 'Accession' number of the same hit, write the 'Accession' number down and copy the following information:
   - Status / Protein Existence
   - Mass (kDa)

Questions (1): - Which target is more similar compared to the original target? - Did you expect this?

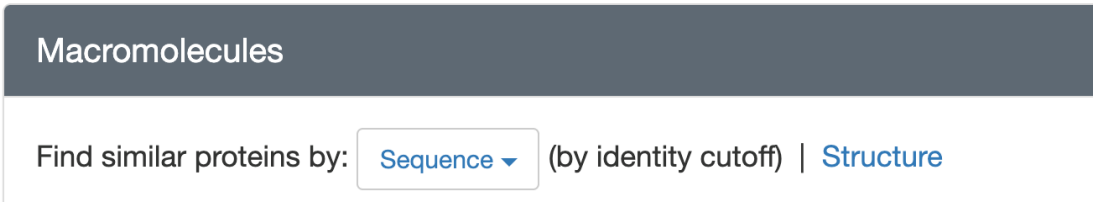
Align the 3 proteins on Uniprot using the 'Accession' numbers.

In the overview, select the following tickboxes: "Similarity, Hydrophobic, Negative, Positive, Aromatic", scroll down and make a screenshot (for report and presentation). Which targets are 'more similar' ?

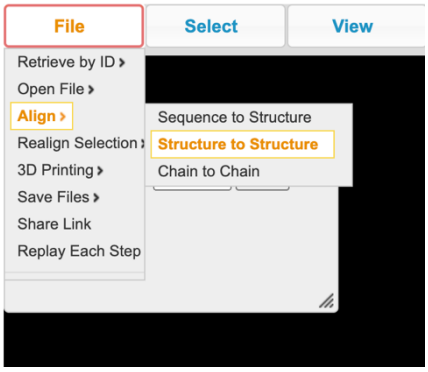
2.0.3 Related proteins (off-targets), based on structure

Since we are in the fortunate position that we have a crystal structure, we are also going to use a 3D similarity search. This is a structural similarity search rather than a sequence similarity search. Now we compare proteins based on their tertiary structure (3D structure)

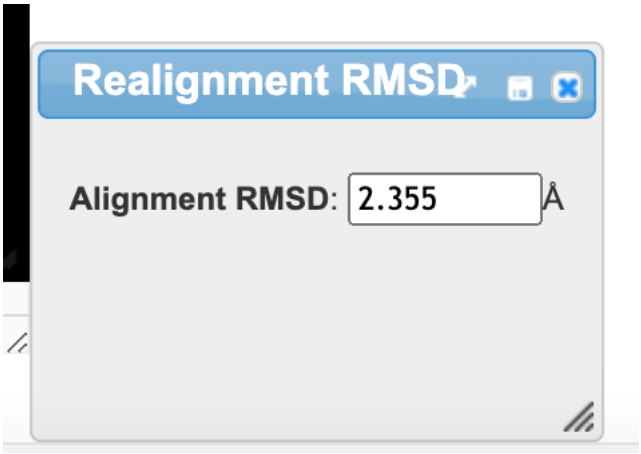
10. Start at http://www.rcsb.org/pdb/home/home.do and find your target using the PDB identifier.
11. Scroll down to 'macromolecules'.
12. Click 'structure'



13. The top one should be your protein. • Write down the accession of the next most similar one • In the case of a GPCR target, ignore the T4Lysozyme hits
14. Go to: https://www.ncbi.nlm.nih.gov/Structure/icn3d/full.html use align and select structure to structure.



15. Input both pdb codes and click all matching molecules superposed.
16. After completion do file realign and select 'on sequence alignment'
17. Select the 'A' chains and click realign. Note the RMSD below (realignment RMSD).



18. Make a screenshot of the superposition.

3 Retrieving a 3D structure

Next, we will prepare our protein. The first step is to download it from the .pdb. We will the ngl viewer for this.

We will add hydrogen atoms to the structure, because these are normally not resolved in the structure (due to limitations in resolution of the experimental method.

19. First, we make a working directory in your home directory:

Next, you need to specify the PDB code of your protein. We have used 4eiy as an example. If you execute the code block below, you will see a viewer with a 3D structure of the protein.

[3]:
```
TARGET_PDB_ID = "2QBS" # Enter your target PDB code here, example = '4eiy'

view = nglview.show_pdbid(TARGET_PDB_ID)
view
```

```
NGLWidget()
```

It is nice to have the overview of the structure, but since we are interested in designing new drugs, it makes more sense to have a closer look at the co-crystallized ligand. For this, we need to know the residue name of the ligand. This is a three later amino acid code that you can retrieve from the RCSB. In this case, the amino acid code is ZMA. First we zoom in on the ligand.

[4]:
```
LIGAND_CODE = "024" # Enter the ligand code here, example = 'zma'

view.center(LIGAND_CODE)
view
```

```
NGLWidget(n_components=1)
```

Next, let's show the residues that are within 5 Angstrom of the ligand

[5]:
```
viewer.show_residues_around(view, selection=LIGAND_CODE)
view
```

```
NGLWidget(n_components=1)
```

Have a look at the residues near the ligand, can you observe any important interactions? Describe in your report which interactions you observe, and what type of interactions they are.

Note that we do not see any hydrogen atoms, do you know why?

In the next stage, we will add the hydrogens and have another look at the structure. We will split the protein and ligand and save them seperately. For this, we use biopython. https://biopython.org/docs/1.75/api/Bio.html

First download the coordinates from RCSB.

[6]:
```
pdbl = PDBList()
pdbl.retrieve_pdb_file(TARGET_PDB_ID, pdir=TARGET_PDB_ID)
```

```
WARNING: The default download format has changed from PDB to PDBx/mmCif

Downloading PDB structure '2QBS'…
```

[6]:
```
'2QBS/2qbs.cif'
```

Next, we generate a BioPython object from the coordinates, which we can use for various tasks.

[7]:
```
parser = MMCIFParser()
structure = parser.get_structure("TARGETPROT",'{}/{}.cif'.
 ↪format(TARGET_PDB_ID,TARGET_PDB_ID.lower()))
```

Now we save the ligand

[8]:
```
class ResSelect(Select):
    def accept_residue(self, residue):
        if residue.get_resname() == LIGAND_CODE:
            return 1
        else:
            return 0

class NonHetSelect(Select):
    def accept_residue(self, residue):
        return 1 if residue.id[0] == " " else 0

io = PDBIO()
io.set_structure(structure)
io.save("ligand-{}.pdb".format(LIGAND_CODE), ResSelect())
io.save("protein-{}.pdb".format(TARGET_PDB_ID), NonHetSelect())
```

Now that we have seperated ligands and protein, we can start adding hydrogens to the protein, we will use LePro for this, which is part of the LeDock program (https://en.wikipedia.org/wiki/LeDock).

21. Prepare the protein:

[9]:
```
command = '../CBR_teaching/bin/lepro protein-{}.pdb'.format(TARGET_PDB_ID)
os.system(command)
shutil.move('pro.pdb','{}_prepped.pdb'.format(TARGET_PDB_ID))
```

[9]:
```
'2QBS_prepped.pdb'
```

[10]:
```
# combine protein and ligand files
filenames = [
'{}_prepped.pdb'.format(TARGET_PDB_ID),
"ligand-{}.pdb".format(LIGAND_CODE)
]
with open('{}-complex.pdb'.format(TARGET_PDB_ID), 'w') as outfile:
    for fname in filenames:
        with open(fname) as infile:
            for line in infile:
                if not "END" in line:
                    outfile.write(line)
```

[11]:
```
with open('{}-complex.pdb'.format(TARGET_PDB_ID)) as f:
    view = nglview.show_file(f, ext="pdb")

view.center(LIGAND_CODE)
viewer.show_residues_around(view, selection=LIGAND_CODE)
view
```

```
NGLWidget()
```

Let's have a look again at the protein