Lecture 22: 3D analysis III

This lecture will cover how to align two proteins two each other to compare their structures

Sequence information

It might seem strange, but getting the sequence of the protein in the pdb is not trivial. You can't use the reference sequence for the given protein (i.e. glc-1's sequence can be downloaded from wormbase.org) because there are often mutations engineered into the protein to make it more amenable to crystallization. There are also often gaps in the protein structure where the residues are flexible/undefined (i.e. if the protein isn't identical in all the crystals you can't resolve the protein. The following will show you how to get the sequence of a pdb using the PPBuilder class. This will return all of the connected residues as a list of polypeptides. You can then use a get_sequence method to get the protein sequence.

```
>>> import Bio.PDB
>>> pdbp = Bio.PDB.PDBParser(QUIET = True)
>>> structure1 = pdbp.get_structure('GluRecA', '3rif_ChainA.pdb')
>>> structure2 = pdbp.get_structure('GluRecB', '3rif_ChainB.pdb')
>>> ppb = Bio.PDB.PPBuilder()

>>> polypeptides1 = ppb.build_peptides(structure1)
>>> polypeptides2 = ppb.build_peptides(structure2)
```

This returns a list of polypeptide objects.

```
>>> polypeptides1
[<Polypeptide start=1 end=340>]
>>> polypeptides2
[<Polypeptide start=1 end=340>]
```

For each polpeptide object, you can use the get_sequence() method to return a Seq object.

```
>>> polypeptides1[0].get_sequence()
Seq('SDSKILAHLFTSGYDFRVRPPTDNGGPVVVSVNMLLRTISKIDVVNMEYSAQLT...FGH')
>>> str(polypeptides1[0].get_sequence())
'SDSKILAHLFTSGYDFRVRPPTDNGGPVVVSVNMLLRTISKIDVVNMEYSAQLTLRESWIDKRLSYGVKGDGQPDFVILTVGHQIWMP
DTFFPNEKQAYKHTIDKPNVLIRIHNDGTVLYSVRISLVLSCPMYLQYYPMDVQQCSIDLASYAYTTKDIEYLWKEHSPLQLKVGLSSS
LPSFQLTNTSTTYCTSVTNTGIYSCLRTTIQLKREFSFYLLQLYIPSCMLVIVSWVSFWFDRTAIPARVTLGVTTLLTMTAQSAGINSQ
LPPVSYIKAIDVWIGACMTFIFCALLEFALVNHIANAGTTEWNDISKRVDLISRALFPVLFFVFNILYWSRFGH'
```

Let's get the sequence for two of the chains ('A' and 'B') and store them as sequence records.

Comparing two protein structures/models

Having the sequence information allows you to align to protein structures together. t is often useful to compare two protein structures. For example, when you have multiple subunits, you might wonder if they have the same shape. Or if you have a protein in its active and inactive shape, you might want to identify the regions of the protein that have moved. There really are a large number of reasons you might want to compare two structures. In order to do this, you need to take into account that the sequence of the two structures might not be identical to each other. So you will need to have some sort of way to identify the residues that you want to compare. You do this by creating a sequence alignment.

```
MNSFSTSAFGPVAFSLGLLLVLPAAFP-APVPPGEDSKDVAAPHRQPLTS
|...|...|.|||| |||:||...||| :.|..|:.::| ..|:| |:.:
MKFLSARDFHPVAF-LGLMLVTTTAFPTSQVRRGDFTED-TTPNR-PVYT

SERIDKQIRYILDGISALRKETCNKSNMCESSKEALAENNLNLPKMAEKD
:.::...|.::|..|..:|||.||.::.|.::.:|||||||.||::...|
TSQVGGLITHVLWEIVEMRKELCNGNSDCMNNDDALAENNLKLPEIQRND

GCFQSGFNEETCLVKIITGLLEFEVYLEYLQNRF-ESSEEQARAVQMSTK
||:|:|:|:|.||:||.:||||:..||||::|.. ::.:::||.:|..|:
GCYQTGYNQEICLLKISSGLLEYHSYLEYMKNNLKDNKKDKARVLQRDTE

VLIQFLQKKAKNLDAITTPDPTTNASLLTKLQAQNQWLQDMTTHLILRSF
.||....::.|:|..|..|.|.:||.|..||::|.:||:..|...||:|.
TLIHIFNQEVKDLHKIVLPTPISNALLTDKLESQKEWLRTKTIQFILKSL

KEFLQSSLRALRQM
:|||:.:||:.||.
EEFLKVTLRSTRQT
```

This allows the structure alignment to compare two independent sequences. The amino acids that are homologous are the amino acids that should be as close together as possible in the structure alignment.

So, the first thing we need to do is to create a sequence alignment between the two proteins using the pairwise2 module in Biopython. This can align two sequences without resorting to writing and reading annoying input and output files.

```
>>> from Bio import pairwise2
```

We also need to specify some parameters for performing the alignment

```
>>> from Bio.Align import substitution_matrices
>>> matrix = substitution_matrices.load(name = 'BLOSUM62')
>>> gap_open = -10
>>> gap_extend = -0.5
```

Finally, we need to convert the sequences into SeqRecord objects that the pairwise2 algorithm expects

```
>>> seq1 = polypeptides1[0].get_sequence()
>>> seq2 = polypeptides2[0].get_sequence()
```

A pairwise alignment is created by finding the alignment that has the highest score. Thus you need a scoring system to tell you how good an alignment is. This is accomplished using 1. A matrix that scores how two amino acids in a given alignment matches. For example, if the two amino acids are identical, then you should get a positive score. If they don't match, you might want to give a negative score. However, it's important to remember that there are certain biochemical similarities between different amino acids that make some differences more tolerable than others. Valine, Leucine, and Isoleucine are all hydrophobic residues and switching between them is often tolerated by the protein.

You also need to give a penalty for opening and extending gaps. The preceding matrix and values are all very reasonable places to start for constructing an alignment.

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
P	-3	-1	-1	7																	P
A	0	1	0	-1	4																A
G	-3	0	-2	-2	0	6															G
N	-3	1	0	-2	-2	0	6														N
D	-3	0	-1	-1	-2	-1	1	6													D
E	-4	0	-1	-1	-1	-2	0	2	5												E
Q	-3	0	-1	-1	-1	-2	0	0	2	5											Q
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8										H
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5									R
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5								K
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5							M
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4					L
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4				V
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6			F
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7		Y
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11	W
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	

(a)

Target sequence	A	L	E	D	N
Functionally Identical or Related Enzymes (property A proteins)	H	L	E	P	L
Sequence-related Proteins (~A proteins)	L	L	P	D	L
Score	-1	0	6	-7	0

(b)

You can now create an alignment using the following command. Note that you are returned a set of alignments and you will need to take the top alignment.

```
>>> alns = pairwise2.align.globalds(seq1,seq2,matrix, gap_open, gap_extend)
>>> top_aln = alns[0]
>>> top_aln
```

```
Alignment(seqA='SDSKILAHLFTSGYDFRVRPPTDNGGPVVVSVNMLLRTISKIDVVNMEYSAQLTLRESWIDKRLSYGVKGDGQ
PDFVILTVGHQIWMPDTFFPNEKQAYKHTIDKPNVLIRIHNDGTVLYSVRISLVLSCPMYLQYYPMDVQQCSIDLASYAYTTKDIEYLW
KEHSPLQLKVGLSSSLPSFQLTNTSTTYCTSVTNTGIYSCLRTTIQLKREFSFYLLQLYIPSCMLVIVSWVSFWFDRTAIPARVTLGVT
TLLTMTAQSAGINSQLPPVSYIKAIDVWIGACMTFIFCALLEFALVNHIANAGTTEWNDISKRVDLISRALFPVLFFVFNILYWSRFGH
',
seqB='SDSKILAHLFTSGYDFRVRPPTDNGGPVVVSVNMLLRTISKIDVVNMEYSAQLTLRESWIDKRLSYGVKGDGQPDFVILTVGH
QIWMPDTFFPNEKQAYKHTIDKPNVLIRIHNDGTVLYSVRISLVLSCPMYLQYYPMDVQQCSIDLASYAYTTKDIEYLWKEHSPLQLKV
GLSSSLPSFQLTNTSTTYCTSVTNTGIYSCLRTTIQLKREFSFYLLQLYIPSCMLVIVSWVSFWFDRTAIPARVTLGVTTLLTMTAQSA
GINSQLPPVSYIKAIDVWIGACMTFIFCALLEFALVNHIANAGTTEWNDISKRVDLISRALFPVLFFVFNILYWSRFGH',
score=1788.0, start=0, end=340)
```

Comparing two structures.

We now have a way to compare two protein sequences. We need to extend this to compare two protein structures using the Structure alignment object. This creates an object that rapidly allows us to take the residue from one chain and identify the homologous residue in the second object. It takes in an MSA object and two protein models:

```
>>> from Bio.Align import MultipleSeqAlignment
>>> alignment =
MultipleSeqAlignment([Bio.SeqRecord.SeqRecord(top_aln[0]),Bio.SeqRecord.SeqRecord
(top_aln[1])])
>>> structure_alignment = Bio.PDB.StructureAlignment(alignment, structure1,
structure2)
>>> structure_alignment
<Bio.PDB.StructureAlignment.StructureAlignment object at 0x10d1228d0>
>>> structure_alignment.duos[0]
(<Residue SER het=  resseq=1 icode= >, <Residue SER het=  resseq=1 icode= >)
>>> a = list(structure_alignment.map12.keys())[0]
>>> a
<Residue VAL het=  resseq=270 icode= >
>>> structure_alignment.map12[a]
<Residue VAL het=  resseq=270 icode= >
```

Aligning two protein structures

We now have everything in place to align two protein objects based upon this comparison using the Superimposer class.

```
>>> sup = Bio.PDB.Superimposer()
>>> type(sup)
<class 'Bio.PDB.Superimposer.Superimposer'>
>>> dir(sup)
['__class__',   '__delattr__',   '__dict__',   '__dir__',   '__doc__',   '__eq__',
'__format__',  '__ge__',  '__getattribute__',  '__gt__',  '__hash__',  '__init__',
'__le__',    '__lt__',    '__module__',    '__ne__',    '__new__',    '__reduce__',
'__reduce_ex__',    '__repr__',    '__setattr__',    '__sizeof__',    '__str__',
'__subclasshook__', '__weakref__', 'apply', 'rms', 'rotran', 'set_atoms']


>>> ref_atoms = []
>>> mov_atoms = []
>>> for duo in structure_alignment.duos:
...     res1 = duo[0]
...     res2 = duo[1]
...     if res1 and res2:
```

```
...         ref_atoms.append(res1['CA'])
...         mov_atoms.append(res2['CA'])


>>> sup.set_atoms(ref_atoms, mov_atoms)
>>> sup.apply(structure2.get_atoms())
>>> io = Bio.PDB.PDBIO()
>>> io.set_structure(structure2)
>>> io.save('Temp.pdb')
```