Class 10 - Structural Bioinformatics (pt. 1)

Lilith Sadil, A16470107

Intro to the RCSB Protein Data Bank (PDB)

The main repository of biomolecular structure is the PDB <www.rcsb.org>.

Let's see what the database contains:

```
stats = read.csv("pdb_stats.csv", row.names=1)
stats
```

	X.ray	EM	NMR	Multiple.methods	Neutron	Other
Protein (only)	161,663	12,592	12,337	200	74	32
Protein/Oligosaccharide	9,348	2,167	34	8	2	0
Protein/NA	8,404	3,924	286	7	0	0
Nucleic acid (only)	2,758	125	1,477	14	3	1
Other	164	9	33	0	0	0
Oligosaccharide (only)	11	0	6	1	0	4

	Total
Protein (only)	186,898
Protein/Oligosaccharide	11,559
Protein/NA	12,621
Nucleic acid (only)	4,378
Other	206
Oligosaccharide (only)	22

> Q1: What percentage of structures in the PDB are solved by X-Ray and Electron Microscopy.

```
# We cannot use: sum(stats$X.ray) since the values under the X.ray column contain commas l

as.numeric(stats$X.ray)
```

```
Warning: NAs introduced by coercion
```

```
[1]  NA  NA  NA  NA 164  11
```

Our first step is to get rid of the commas in the dataset. We can use `gsub(pattern, replacement, x)` which stands for global substitution:

```
x=stats$X.ray
sum(as.numeric(gsub(",", "", x)))
```

```
[1] 182348
```

```
#Here, we replaced every instance of a comma in x with nothing (i.e. deletes the commas)
```

Now, we can turn this code snippet into a function in order to convert all the data in the table into numbers without commas:

```
sumcomma =  function(x){
  sum(as.numeric(gsub(",", "", x))) #We can set x to any column in "stats" in order to cal
}
```

```
sumcomma(stats$Total)
```

```
[1] 215684
```

```
#sumcomma(stats$Total) gives us the sum of the Total column in stats
```

To do our calculation column by column, we could do (sumcomma(stats$X.ray$)/$sumcomma($stats$Total$)). Instead, we can use "apply" to run this calculation across all columns at once:

```
apply(stats, 2, sumcomma)
```

```
     X.ray         EM        NMR Multiple.methods
    182348      18817      14173              230
   Neutron      Other      Total
        79         37     215684
```

Next, we can divide every column by the sum of the Total column, turning every column into a percent:

```
apply(stats, 2, sumcomma) / sumcomma(stats$Total)
```

```
        X.ray                EM               NMR Multiple.methods
0.8454405519     0.0872433746     0.0657118748     0.0010663749
     Neutron            Other            Total
0.0003662766     0.0001715473     1.0000000000
```

From this, we can see that 84.5% of structures in the PDB are solved by X-Ray and an additional 8.7% are solved by Electron Microscopy.

```
(186898/248895733) * 100
```

```
[1] 0.07509088
```

Only 7% of the HIV-1 protease sequences we know have structures in the PDB

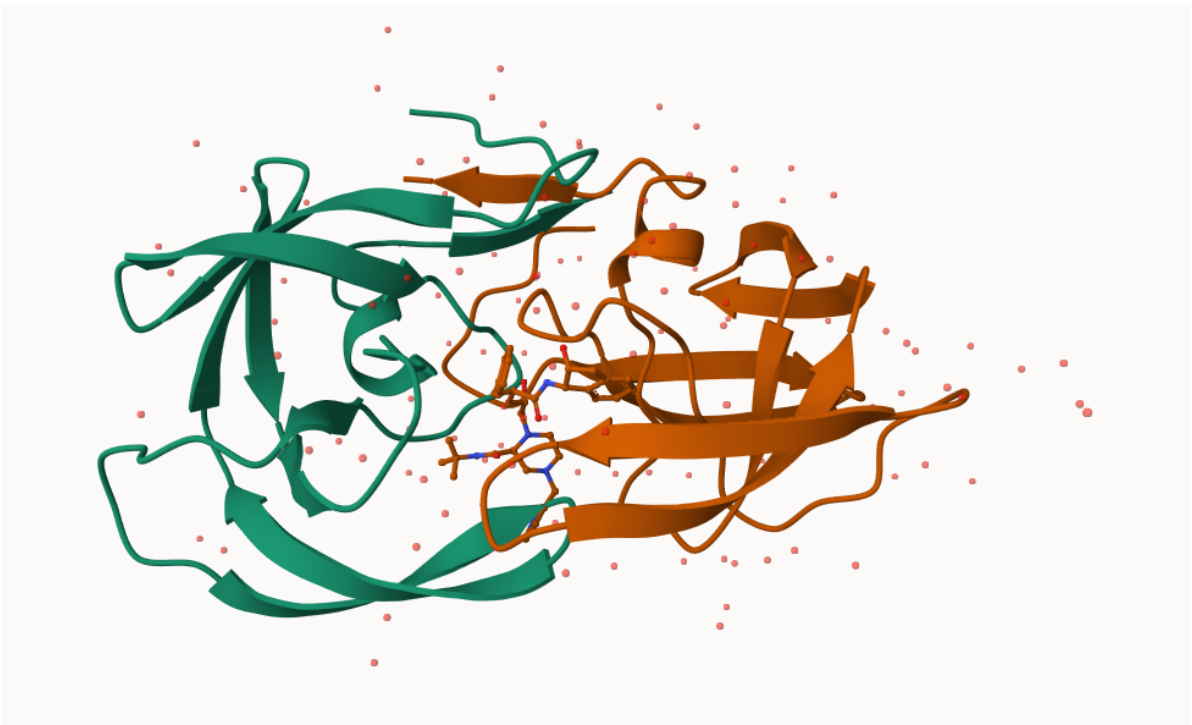
Visualizing the HIV-1 Protease Structure

Mol* ("mol-star") viewer is now everywhere.

> Q4: Water molecules normally have 3 atoms. Why do we see just one atom per water molecule in this structure?

Our resolution is set to 2 angstroms; in order to see hydrogen, we would need to increase the resolution to 1 angstrom

If we want to insert our image from Mol* into our document, we use the camera icon to take and download a picture, move the image into our project folder for R, and then insert the image name into the syntax below:



In the image below, the 1HSG protein is shown with the two ASP25 residues (one on each homodimer) and the critical central water molecule highlighted

Introduction to Bio3D in R

```
library(bio3d)
```

```
pdb = read.pdb("1hsg")
```

```
 Note: Accessing on-line PDB file
```

```
pdb
```

```
Call:  read.pdb(file = "1hsg")

  Total Models#: 1
    Total Atoms#: 1686,  XYZs#: 5058  Chains#: 2  (values: A B)

    Protein Atoms#: 1514  (residues/Calpha atoms#: 198)
```

```
     Nucleic acid Atoms#: 0  (residues/phosphate atoms#: 0)

     Non-protein/nucleic Atoms#: 172  (residues: 128)
     Non-protein/nucleic resid values: [ HOH (127), MK1 (1) ]

   Protein sequence:
      PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIGGFIKVRQYD
      QILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFPQITLWQRPLVTIKIGGQLKE
      ALLDTGADDTVLEEMSLPGRWKPKMIGGIGGFIKVRQYDQILIEICGHKAIGTVLVGPTP
      VNIIGRNLLTQIGCTLNF

+ attr: atom, xyz, seqres, helix, sheet,
        calpha, remark, call
```

```
head(pdb$atom)
```

```
  type eleno elety  alt resid chain resno insert      x      y     z o     b
1 ATOM     1     N <NA>   PRO     A     1   <NA> 29.361 39.686 5.862 1 38.10
2 ATOM     2    CA <NA>   PRO     A     1   <NA> 30.307 38.663 5.319 1 40.62
3 ATOM     3     C <NA>   PRO     A     1   <NA> 29.760 38.071 4.022 1 42.64
4 ATOM     4     O <NA>   PRO     A     1   <NA> 28.600 38.302 3.676 1 43.40
5 ATOM     5    CB <NA>   PRO     A     1   <NA> 30.508 37.541 6.342 1 37.87
6 ATOM     6    CG <NA>   PRO     A     1   <NA> 29.296 37.591 7.162 1 38.40
  segid elesy charge
1  <NA>     N   <NA>
2  <NA>     C   <NA>
3  <NA>     C   <NA>
4  <NA>     O   <NA>
5  <NA>     C   <NA>
6  <NA>     C   <NA>
```

```
#Tells us the atoms in the structure & their positions (ex. CA = Carbon at position alpha
```

```
pdbseq(pdb)[25]
```

```
 25
"D"
```

```
#Tells us that the 25th residue in the structure is "D', Asp
```

Predicting functional motions of a single structure

We can do a bioinformatics prediction of functional motions (i.e. flexibility/dynamics):

```
pdb2=read.pdb("6s36")
```

```
  Note: Accessing on-line PDB file
   PDB has ALT records, taking A only, rm.alt=TRUE
```

```
pdb2
```

```
 Call:  read.pdb(file = "6s36")

   Total Models#: 1
     Total Atoms#: 1898,  XYZs#: 5694  Chains#: 1  (values: A)

     Protein Atoms#: 1654  (residues/Calpha atoms#: 214)
     Nucleic acid Atoms#: 0  (residues/phosphate atoms#: 0)

     Non-protein/nucleic Atoms#: 244  (residues: 244)
     Non-protein/nucleic resid values: [ CL (3), HOH (238), MG (2), NA (1) ]

   Protein sequence:
      MRIILLGAPGAGKGTQAQFIMEKYGIPQISTGDMLRAAVKSGSELGKQAKDIMDAGKLVT
      DELVIALVKERIAQEDCRNGFLLDGFPRTIPQADAMKEAGINVDYVLEFDVPDELIVDKI
      VGRRVHAPSGRVYHVKFNPPKVEGKDDVTGEELTTRKDDQEETVRKRLVEYHQMTAPLIG
      YYSKEAEAGNTKYAKVDGTKPVAEVRADLEKILG

+ attr: atom, xyz, seqres, helix, sheet,
        calpha, remark, call
```

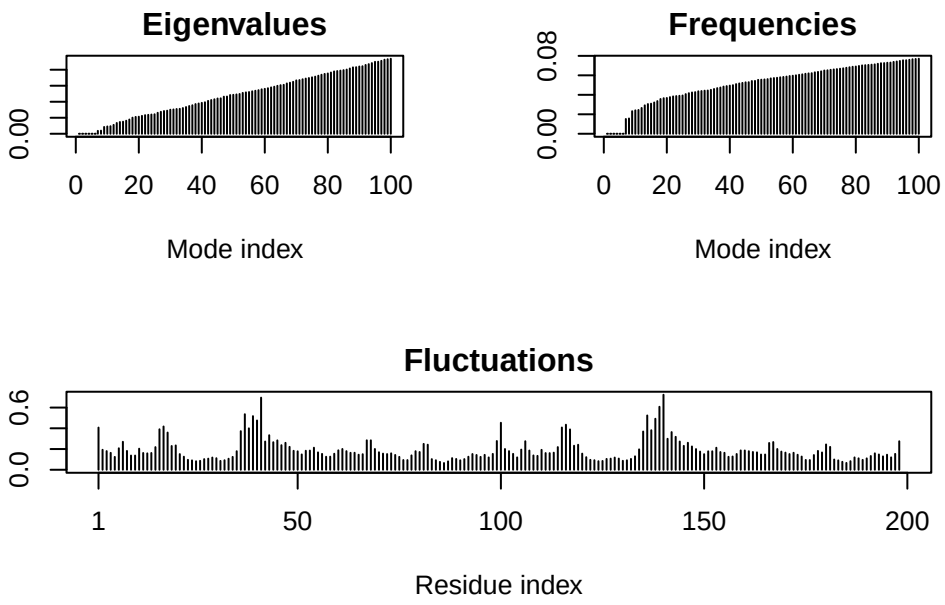
```
m = nma(pdb) # nma stands for normal mode analysis; joins all amino acids with "springs" a
```

```
Warning in nma.pdb(pdb): Possible multi-chain structure or missing in-structure residue(s) pr
  Fluctuations at neighboring positions may be affected.

 Building Hessian...        Done in 0.027 seconds.
 Diagonalizing Hessian...   Done in 0.584 seconds.
```

```
m
```

```
Call:
  nma.pdb(pdb = pdb)

Class:
  VibrationalModes (nma)

Number of modes:
  594 (6 trivial)

Frequencies:
  Mode 7:   0.015
  Mode 8:   0.016
  Mode 9:   0.023
  Mode 10:  0.024
  Mode 11:  0.024
  Mode 12:  0.026

+ attr: modes, frequencies, force.constants, fluctuations,
        U, L, xyz, mass, temp, triv.modes, natoms, call
```

```
plot(m)
```




```
#plotting the normal mode analysis shows that there are certain regions of the protein tha
```

We can use the following code to create a trajectory plot of the protein's movement. This file gets saved into our R files. We can then import it to Mol* & watch the animation run.

```
mktrj(m, file="adk_m7.pdb")
```