Class10

Kalisa Kang (PID A16741690)

What is in the PDB database?

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

Let's see what this database contains:

```
stats <- read.csv("Data Export Summary.csv", row.names=1)
stats
```

	X.ray	EM	NMR	Multiple.methods	Neutron	Other
Protein (only)	163,468	13,582	12,390	204	74	32
Protein/Oligosaccharide	9,437	2,287	34	8	2	0
Protein/NA	8,482	4,181	286	7	0	0
Nucleic acid (only)	2,800	132	1,488	14	3	1
Other	164	9	33	0	0	0
Oligosaccharide (only)	11	0	6	1	0	4

	Total
Protein (only)	189,750
Protein/Oligosaccharide	11,768
Protein/NA	12,956
Nucleic acid (only)	4,438
Other	206
Oligosaccharide (only)	22

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

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

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

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

We need to get rid of the commas. Can you find a function to do this? `gsub()` can be used. The g stands for global and sub stands for substitution (ie substitute all).

sub(pattern, replacement, x) but they're still around quotes meaning they're characters, so use `as.numeric()` to make them numbers we can do math on.

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

```
[1] 184362
```

Apply to all columns now by turning the previous code into a function: use `apply()` to work on the entire table of data.

```
sumcomma <- function(x) {
  sum( as.numeric(gsub(",", "", x)) )
}

sumcomma(stats$X.ray)
```

```
[1] 184362
```

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

```
[1] 219140
```

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

```
        X.ray                EM               NMR Multiple.methods
0.8412978005     0.0921374464     0.0649676006     0.0010678105
     Neutron            Other            Total
0.0003605001     0.0001688418     1.0000000000
```

84.13% of PDB structures are solved by X-ray and 9.21% are solved by EM.

```
n.total <- sumcomma(stats$Total)
n.total
```

```
[1] 219140
```

```
sumcomma(stats$EM)
```

```
[1] 20191
```

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

```
          X.ray              EM             NMR Multiple.methods
         184362           20191           14237             234
        Neutron           Other           Total
             79              37          219140
```

Q2: What proportion of structures in the PDB are protein?

```
n.protein <- sumcomma(stats[1, "Total"])
n.protein
```

```
[1] 189750
```

```
n.protein/n.total
```

```
[1] 0.8658848
```

86.59% of structures in the PDB are protein.

In UniProt, there are 248,805,733 entries, which compared to the PDB protein entries (189750), means that there are only ~7% of known sequences with a known structure.

> Q3: Type HIV in the PDB website search box on the home page and determine how many HIV-1 protease structures are in the current PDB?

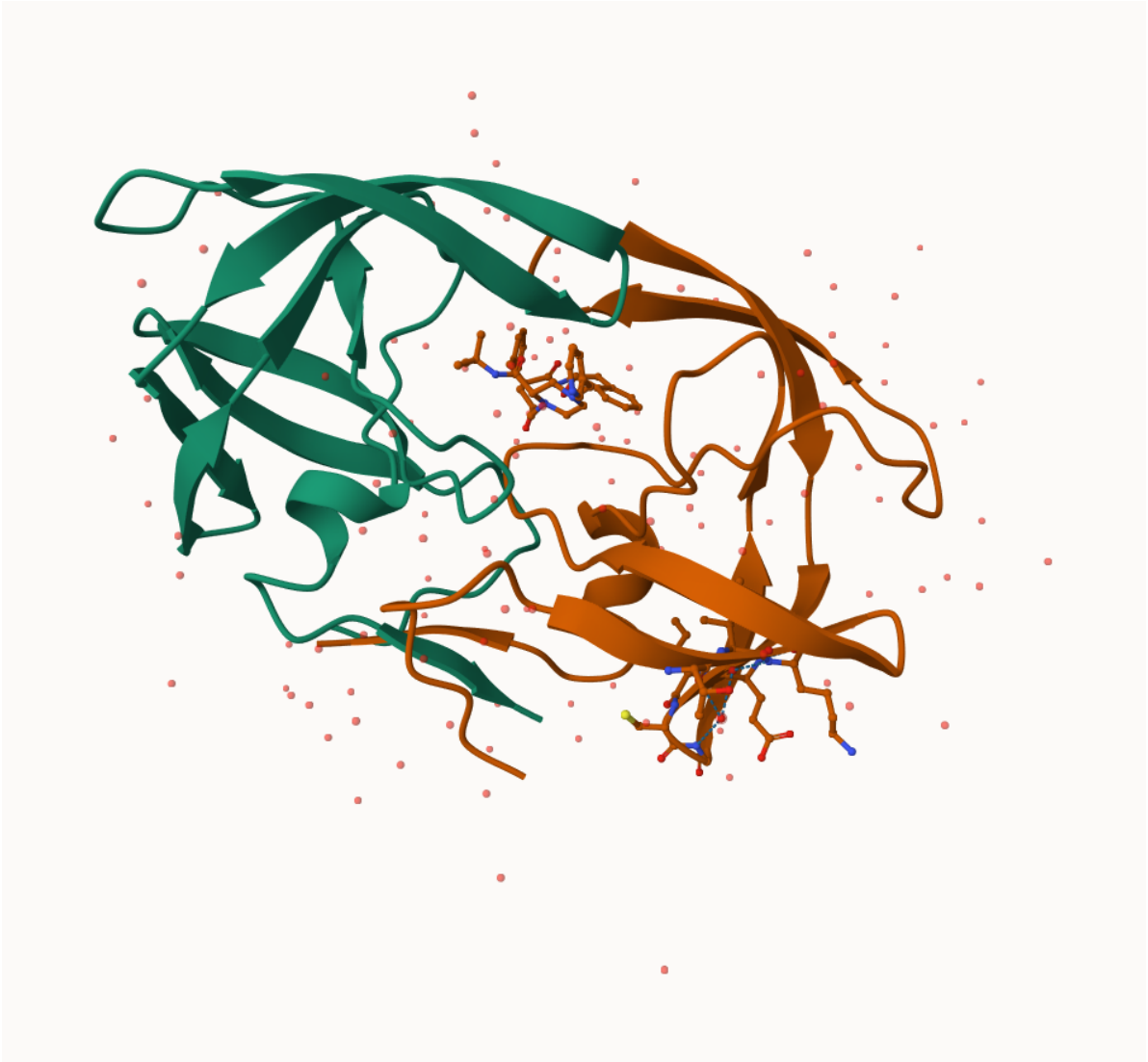


Figure 1: My first molecular image

Visualizing the HIV-1 protease structure

Mol* ("mol-star") viewer is now ubiquitous. The homepage link is here: https://molstar.org/viewer/

I want to insert my image from Mol* here.

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

Hydrogen is not shown because the atom is smaller than the resolution level.

> Q5: There is a critical "conserved" water molecule in the binding site. Can you identify this water molecule? What residue number does this water molecule have

The water molecule has 308 residue number.

Working with bio3d package.

```
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>
```

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

```
 25
"D"
```

Predicting functional motions of a single structure

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

```
pdb <- read.pdb("6s36")
```

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

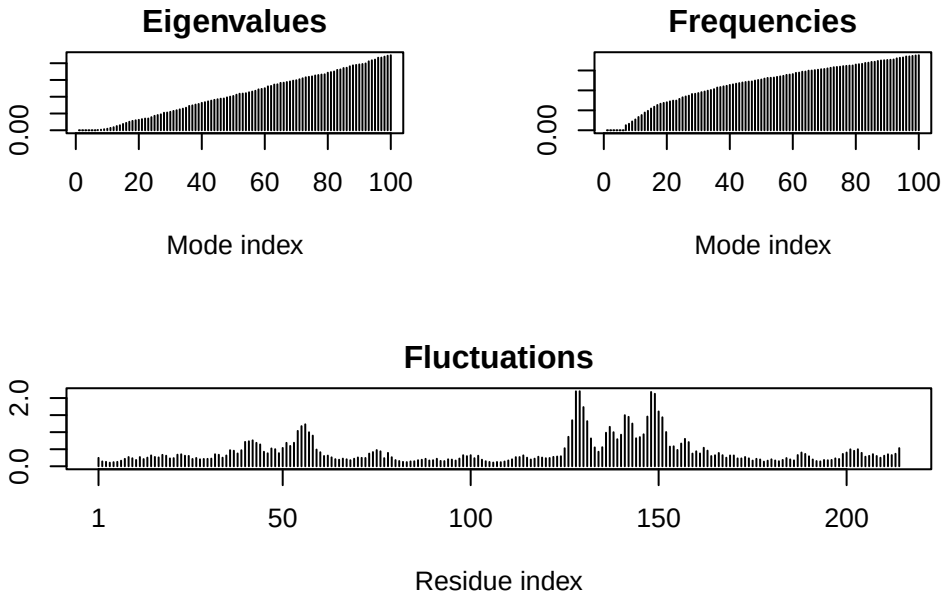
```
pdb
```

```
 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)
```

```
Building Hessian...         Done in 0.015 seconds.
Diagonalizing Hessian...    Done in 0.272 seconds.
```

```
plot(m)
```




```
# Normal mode analysis
```

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