Class 10 Structural Bioinformatics pt1

Vanesa Fernandez PID A59026769

What is the PDB database anyway?

I grabbed summary data from:

https://www.rcsb.org/stats/summary

```
pdbstats <- read.csv("pdb_stats.csv", row.names = 1)
pdbstats
```

```
                         X.ray     EM    NMR Multiple.methods Neutron Other
Protein (only)         167,317 15,698 12,534              208      77    32
Protein/Oligosaccharide  9,645  2,639     34                8       2     0
Protein/NA               8,735  4,718    286                7       0     0
Nucleic acid (only)      2,869    138  1,507               14       3     1
Other                      170     10     33                0       0     0
Oligosaccharide (only)      11      0      6                1       0     4
                         Total
Protein (only)         195,866
Protein/Oligosaccharide 12,328
Protein/NA              13,746
Nucleic acid (only)      4,532
Other                      213
Oligosaccharide (only)      22
```

##The results in the table is telling us the type of data , meaning that these are characters...

```
pdbstats$Total
```

```
[1] "195,866" "12,328"  "13,746"  "4,532"   "213"     "22"
```

##… Thus we need to transform the data to numerical items. We can use “claude.ai”

```
x <- pdbstats$Total
x
```

```
[1] "195,866" "12,328"  "13,746"  "4,532"   "213"     "22"
```

```
#as.numeric()
gsub(',','****',x)
```

```
[1] "195****866" "12****328"  "13****746"  "4****532"   "213"
[6] "22"
```

```
convert_comma_numbers <- function(x) {
  #remove commas
  x <- gsub(',','',x)

  #convert to numeric
  x <- as.numeric (x)

  return (x)
}
```

```
#convert_comma_numbers(pdbstats$Total)

n.tot <- sum(convert_comma_numbers(pdbstats$Total))
n.tot
```

```
[1] 226707
```

```
 n.xray <- sum(convert_comma_numbers(pdbstats$X.ray))
 n.em <- sum(convert_comma_numbers(pdbstats$EM))
```

```
n.xray/n.tot*100
```

```
[1] 83.25592
```

```
n.em/n.tot*100
```

```
[1] 10.2348
```

```
apply(pdbstats, 2, convert_comma_numbers)
```

```
      X.ray    EM   NMR Multiple.methods Neutron Other  Total
[1,] 167317 15698 12534               208      77    32 195866
[2,]   9645  2639    34                 8       2     0  12328
[3,]   8735  4718   286                 7       0     0  13746
[4,]   2869   138  1507                14       3     1   4532
[5,]    170    10    33                 0       0     0    213
[6,]     11     0     6                 1       0     4     22
```

```
195866/ 248838887
```

```
[1] 0.0007871197
```

```
colSums(apply(pdbstats,2,convert_comma_numbers)) / n.tot
```

```
          X.ray              EM             NMR Multiple.methods
   0.8325592064    0.1023479646    0.0635181093    0.0010498132
        Neutron           Other           Total
   0.0003617003    0.0001632063    1.0000000000
```

The 'apply()' function is very useful as it can take any function and apply " it over either the ROWS or COLS of a data.frame.

```
rowSums(apply(pdbstats, 1, convert_comma_numbers))
```

```
[1] 188747  23203  14400    238     82     37 226707
```

"2" is the row

```
#apply(data, 2, FUN)
```

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

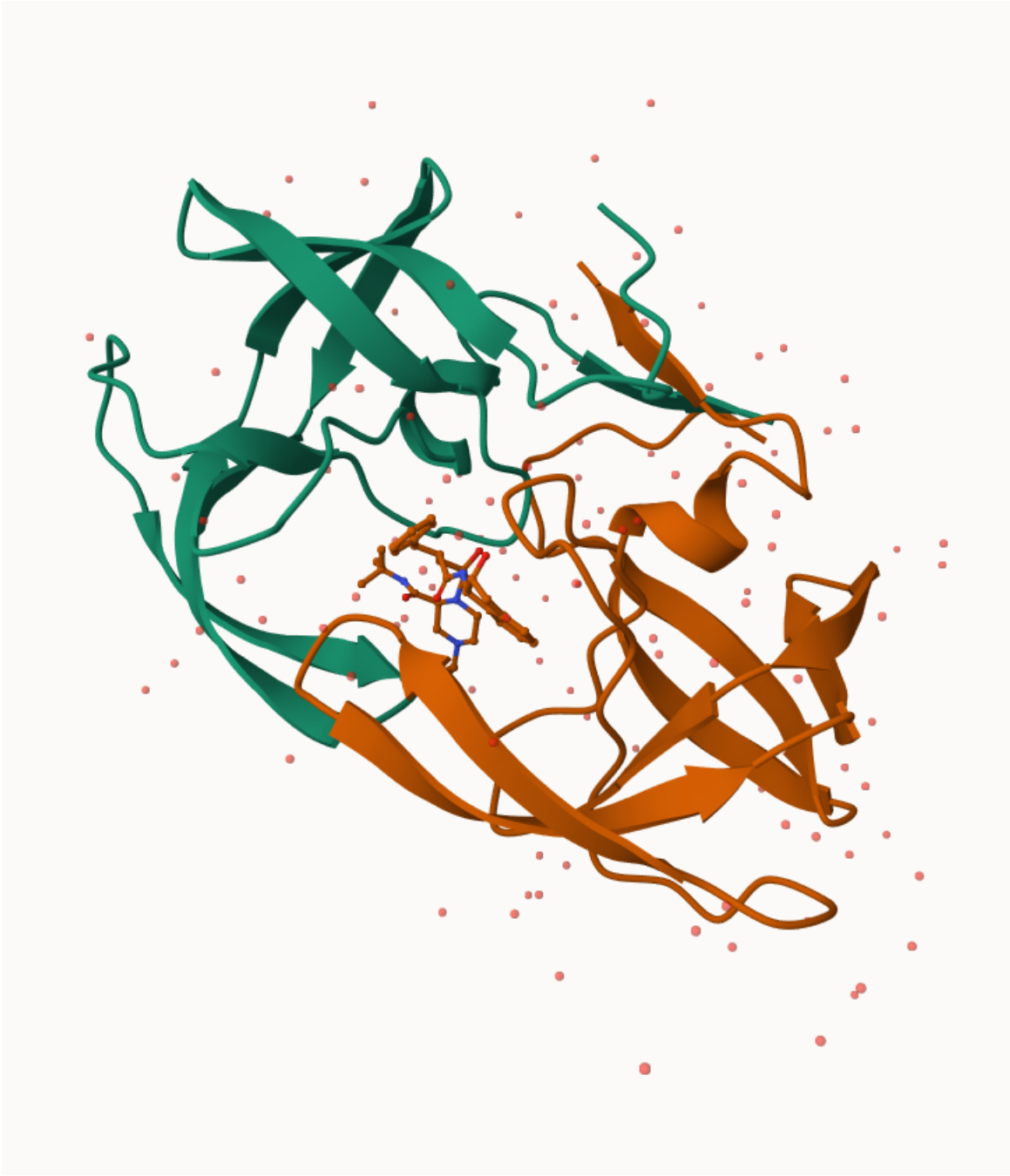
A: Only 10.23% of the structures

```
##install.packages("readr")

library (readr)
read_csv("pdb_stats.csv")
```

```
Rows: 6 Columns: 8
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): Molecular Type
dbl (3): Multiple methods, Neutron, Other
num (4): X-ray, EM, NMR, Total

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

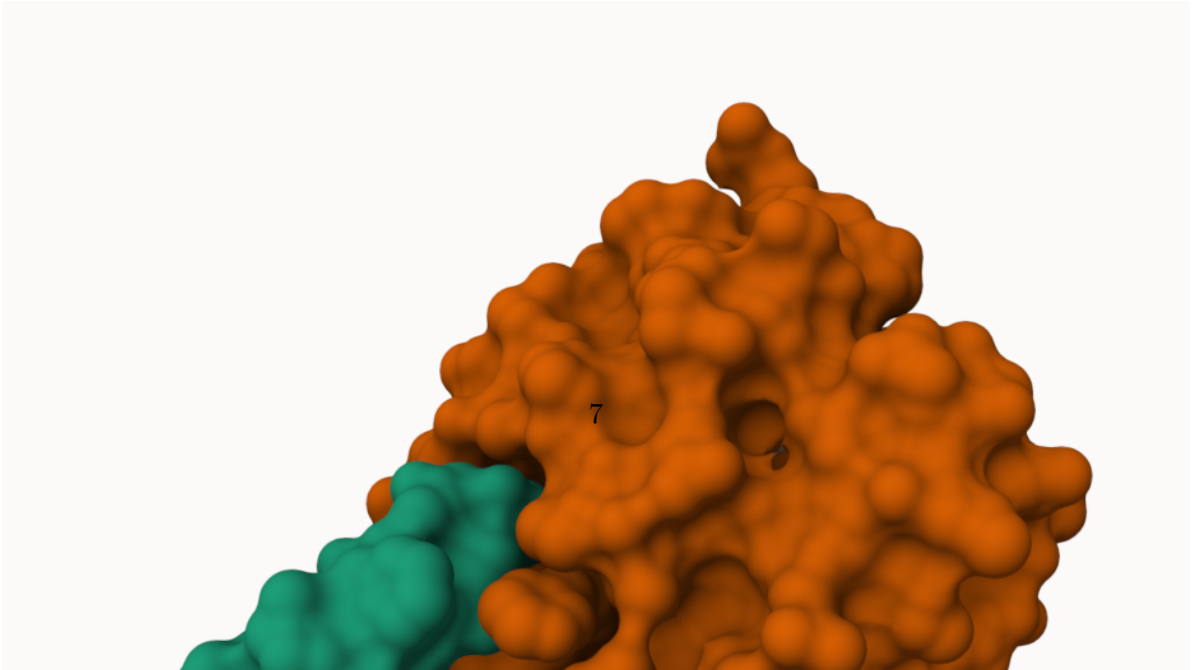
```
# A tibble: 6 x 8
  `Molecular Type`   `X-ray`    EM   NMR `Multiple methods` Neutron Other  Total
  <chr>                <dbl> <dbl> <dbl>              <dbl>   <dbl> <dbl>  <dbl>
1 Protein (only)      167317 15698 12534                208      77    32 195866
2 Protein/Oligosacc~    9645  2639    34                  8       2     0  12328
3 Protein/NA            8735  4718   286                  7       0     0  13746
4 Nucleic acid (onl~    2869   138  1507                 14       3     1   4532
5 Other                  170    10    33                  0       0     0    213
6 Oligosaccharide (~      11     0     6                  1       0     4     22
```

Using Mol*



https://molstar.org

Another cool image, not as good as Barry's, but still I am happy with my creation



Bio3D package

```
#install.packages("bio3d")

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
```

```
attributes(pdb)
```

```
$names
[1] "atom"   "xyz"    "seqres" "helix"  "sheet"  "calpha" "remark" "call"

$class
[1] "pdb" "sse"
```

```
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"
```

Q7. How many amino acid residues are there in this pdb object (structure)

```
length(pdbseq(pdb))
```

```
[1] 198
```

Functional dynamics prediction

Predicting functional motions of a single structure

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

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

```
adk
```

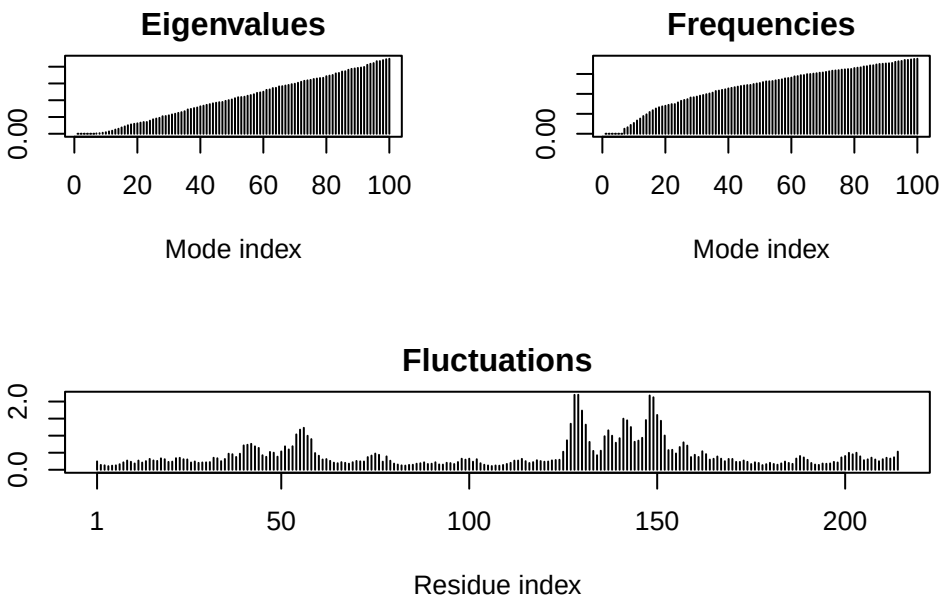
```
 Call:  read.pdb(file = "6s36")

   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
```

```
#install.packages("r3dmol")
#install.packages("bio3d")
#install.packages("shiny")

source("https://tinyurl.com/viewpdb")

library (r3dmol)

#view.pdb(pdb)
```

```
modes <- nma(adk)
```

```
 Building Hessian...        Done in 0.052 seconds.
 Diagonalizing Hessian...   Done in 0.233 seconds.
```

```
plot(modes)
```



We uploaded the "adk.pdb" file to molstar website...

```
mktrj(modes, file="adk.pdb")
```

##... and are able to see the animation (mathematical calculation) of one single structure from the protein Open Files» Select » Apply. You can also export the animation for a presentation.

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

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/rp/1tr_jlnx13d1lfgh27qq4qk40000gq/T//RtmpTsaCNO/6s36.pdb exists.
Skipping download
```

```
   PDB has ALT records, taking A only, rm.alt=TRUE
```

```
modes <- nma(adk)
```

```
 Building Hessian...        Done in 0.047 seconds.
 Diagonalizing Hessian...   Done in 0.232 seconds.
```

```
mktrj(modes, pdb=adk) ## this part give us the animation
mktrj(modes, file= "adk.pdb")
```

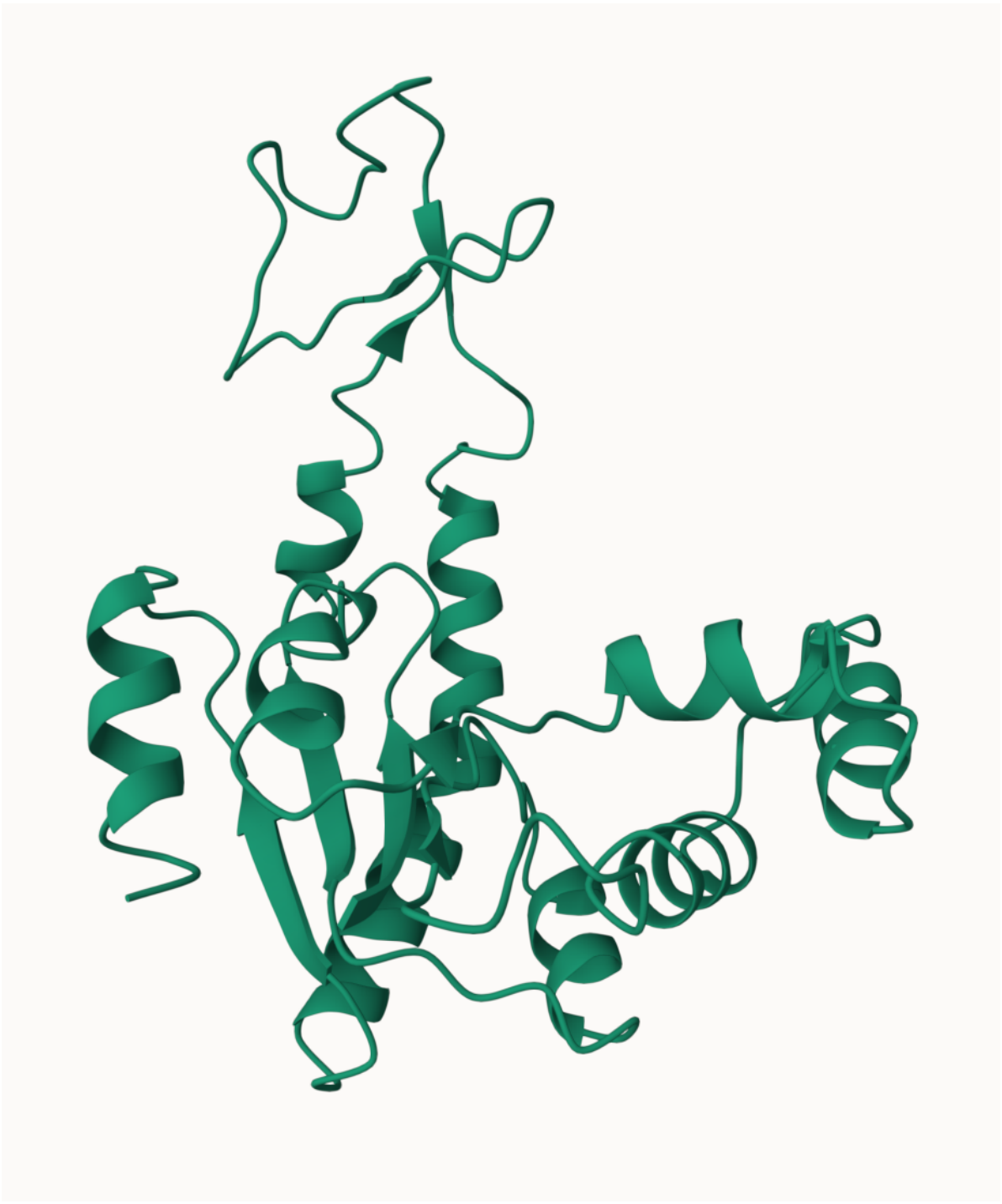


Figure 1: This is a screenshot of my animation