Class 09: Structural Bioinformatics

Divya Shetty (A15390408)

2/16/2022

Intro to RCSB Protein Data Bank

PDB Statistics

Examine the CSV file taken from the PDB site.

```
pdb <- read.csv("Data_Export_Summary.csv", row.names = 1)
pdb
```

```
##                          X.ray   NMR   EM Multiple.methods Neutron Other  Total
## Protein (only)          144433 11881 6732              182      70    32 163330
## Protein/Oligosaccharide   8543    31 1125                5       0     0   9704
## Protein/NA                7621   274 2165                3       0     0  10063
## Nucleic acid (only)       2396  1399   61                8       2     1   3867
## Other                      150    31    3                0       0     0    184
## Oligosaccharide (only)      11     6    0                1       0     4     22
```

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

```
totals <- colSums(pdb)
totals/totals["Total"] * 100
```

```
##            X.ray              NMR               EM Multiple.methods
##      87.16888390       7.27787573       5.38868408       0.10632046
##          Neutron            Other            Total
##       0.03846770       0.01976813     100.00000000
```

About 87.17% of structures in the PDB are solved by X-Ray and 5.39% by Electron Microscopy.

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

```
#proportion
pdb$Total[1] / sum(pdb$Total)
```

```
## [1] 0.8726292
```

About 87.26% of the PDB structures are proteins.

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?

In the current PDB, there are 860 HIV-1 protease structures.

Visualizing the HIV-1 Protease Structure

Using Atom Selection

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

In the structure, only the oxygen atoms are visualized, with the two hydrogen atoms not being included. This means there will only be one atom for each water molecule.

Q5: There is a conserved water molecule in the binding site. Can you identify this water molecule? What residue number does this water molecule have (see note below)?

The residue number is HOH332.

Optional: Generate and save a figure clearly showing the two distinct chains of HIV-protease along with the ligand.

VMD rendered image of 1HSG

Discussion Topic: Can you think of a way in which indinavir, or even larger ligands and substrates, could enter the binding site?

Intro to Bio3D in R

Reading PDB File Data into R

Load the Bio3D package!

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

Read a PDB file into R.

```
hsg <- read.pdb("1hsg")
```

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

```
hsg
```

```
## 
##  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
```

Q7: How many amino acid residues are there in this pdb object?

There are 198 amino acid residues in this object.

Q8: Name one of the two non-protein residues?

One of the non-protein residues is HOH, known as water.

Q9: How many protein chains are in this structure?

There are 2 protein chains.

Examine some of the attributes of the PDB object.

```
attributes(hsg)
```

```
## $names
## [1] "atom"   "xyz"    "seqres" "helix"  "sheet"  "calpha" "remark" "call"  
## 
## $class
## [1] "pdb" "sse"
```

```
head(hsg$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>
```

Comparative Structure Analysis of Adenylate Kinase

Set-up

Install the following in the R console.

```
#install.packages("ggrepel")
#install.packages("devtools")
#install.packages("BiocManager")

#BiocManager::install("msa")
#devtools::install_bitbucket("Grantlab/bio3d-view")
```

Q10. Which of the packages above is found only on BioConductor and not CRAN?

The “msa” package is found only on BioConductor.

Q11. Which of the above packages is not found on BioConductor or CRAN?

The “bio3d-view” package isn’t found on either BioConductor or CRAN.

Q12. True or False? Functions from the devtools package can be used to install packages from GitHub and BitBucket?

TRUE.

Search & Retrieve ADK Structures

First, we need to find the sequence of chain A of 1AKE.

```
aa <- get.seq("1ake_A")
```

```
## Warning in get.seq("1ake_A"): Removing existing file: seqs.fasta
```

```
## Fetching... Please wait. Done.
```

```
aa
```

```
##              1        .         .         .         .         .         60
## pdb|1AKE|A   MRIILLGAPGAGKGTQAQFIMEKYGIPQISTGDMLRAAVKSGSELGKQAKDIMDAGKLVT
##              1        .         .         .         .         .         60
##
##             61        .         .         .         .         .         120
## pdb|1AKE|A   DELVIALVKERIAQEDCRNGFLLDGFPRTIPQADAMKEAGINVDYVLEFDVPDELIVDRI
##             61        .         .         .         .         .         120
##
##            121        .         .         .         .         .         180
## pdb|1AKE|A   VGRRVHAPSGRVYHVKFNPPKVEGKDDVTGEELTTRKDDQEETVRKRLVEYHQMTAPLIG
##            121        .         .         .         .         .         180
##
##            181        .         .         .   214
## pdb|1AKE|A   YYSKEAEAGNTKYAKVDGTKPVAEVRADLEKILG
##            181        .         .         .   214
##
## Call:
##   read.fasta(file = outfile)
##
## Class:
##   fasta
##
## Alignment dimensions:
##   1 sequence rows; 214 position columns (214 non-gap, 0 gap)
##
## + attr: id, ali, call
```

Q13. How many amino acids are in this sequence, i.e. how long is this sequence?

There are 214 amino acids in the sequence.

Now we can use this sequence to BLAST search the PDB database to find similar sequences.

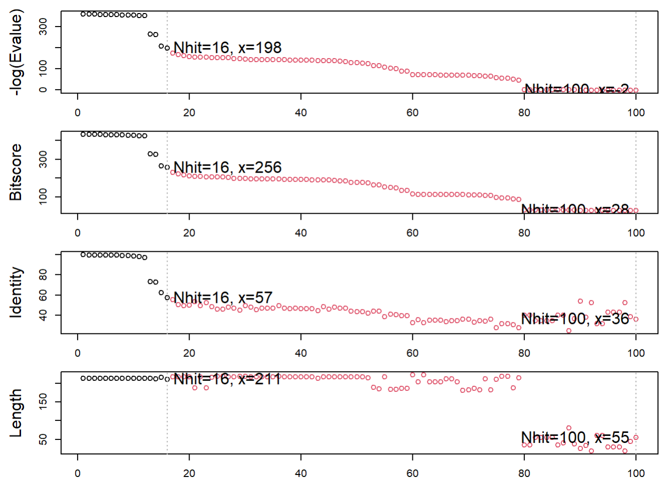
```
aa_blast <- blast.pdb(aa)
```

```
##  Searching ... please wait (updates every 5 seconds) RID = 11BYJ36J013
##  ...
##  Reporting 100 hits
```

We can visualize and filter the BLAST results using the function plot.blast().

```
hits <- plot.blast(aa_blast)
```

```
##   * Possible cutoff values:    197 -3
##             Yielding Nhits:    16 100
##
##   * Chosen cutoff value of:    197
##             Yielding Nhits:    16
```



Here, we can see the top scoring hits from the BLAST results.

```
head(hits$pdb.id)
```

```
## [1] "1AKE_A" "4X8M_A" "6S36_A" "6RZE_A" "4X8H_A" "3HPR_A"
```

With the above information, we can use get.pdb() to fetch and parse the dientified structures.

```
# Download releated PDB files
files <- get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE)
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 1AKE.pdb exists. Skipping download
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 4X8M.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 6S36.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 6RZE.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 4X8H.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 3HPR.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 1E4V.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 5EJE.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 1E4Y.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 3X2S.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 6HAP.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 6HAM.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 4K46.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 4NP6.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 3GMT.pdb exists. Skipping download
```

```
## Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE): pdbs/
## 4PZL.pdb exists. Skipping download
```

```
## 
  |                                                                            
  |                                                                      |   0%
  |                                                                            
  |====                                                                  |   6%
  |                                                                            
  |=========                                                             |  12%
  |                                                                            
  |=============                                                         |  19%
  |                                                                            
  |==================                                                    |  25%
  |                                                                            
  |======================                                                |  31%
  |                                                                            
  |==========================                                            |  38%
  |                                                                            
  |===============================                                       |  44%
  |                                                                            
  |===================================                                   |  50%
  |                                                                            
  |=======================================                               |  56%
  |                                                                            
  |============================================                          |  62%
  |                                                                            
  |================================================                      |  69%
  |                                                                            
  |====================================================                  |  75%
  |                                                                            
  |=========================================================             |  81%
  |                                                                            
  |=============================================================         |  88%
  |                                                                            
  |==================================================================    |  94%
  |                                                                            
  |======================================================================| 100%
```

Align and Superpose Structures

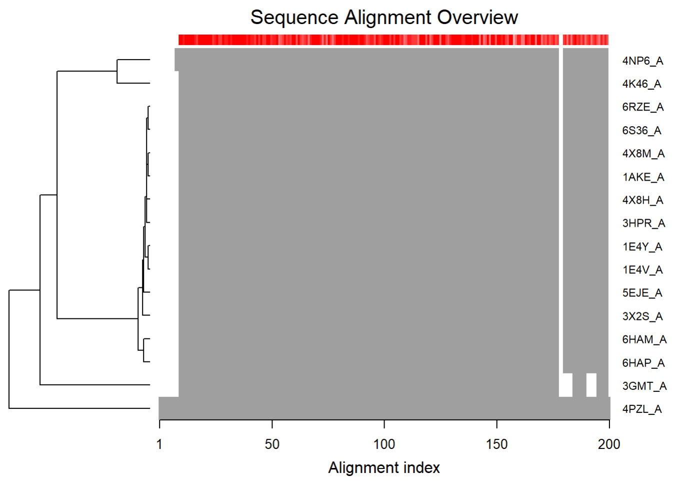
The code below will align and fit the identified PDB structures.

```
#align PDBs
pdbs <- pdbaln(files, fit = TRUE, exefile = "msa")
```

```
## Reading PDB files:
## pdbs/split_chain/1AKE_A.pdb
## pdbs/split_chain/4X8M_A.pdb
## pdbs/split_chain/6S36_A.pdb
## pdbs/split_chain/6RZE_A.pdb
## pdbs/split_chain/4X8H_A.pdb
## pdbs/split_chain/3HPR_A.pdb
## pdbs/split_chain/1E4V_A.pdb
## pdbs/split_chain/5EJE_A.pdb
## pdbs/split_chain/1E4Y_A.pdb
## pdbs/split_chain/3X2S_A.pdb
## pdbs/split_chain/6HAP_A.pdb
## pdbs/split_chain/6HAM_A.pdb
## pdbs/split_chain/4K46_A.pdb
## pdbs/split_chain/4NP6_A.pdb
## pdbs/split_chain/3GMT_A.pdb
## pdbs/split_chain/4PZL_A.pdb
##    PDB has ALT records, taking A only, rm.alt=TRUE
## ..   PDB has ALT records, taking A only, rm.alt=TRUE
## .   PDB has ALT records, taking A only, rm.alt=TRUE
## ..   PDB has ALT records, taking A only, rm.alt=TRUE
## ..   PDB has ALT records, taking A only, rm.alt=TRUE
## ....   PDB has ALT records, taking A only, rm.alt=TRUE
## .   PDB has ALT records, taking A only, rm.alt=TRUE
## ....
##
## Extracting sequences
##
## pdb/seq: 1   name: pdbs/split_chain/1AKE_A.pdb
##    PDB has ALT records, taking A only, rm.alt=TRUE
## pdb/seq: 2   name: pdbs/split_chain/4X8M_A.pdb
## pdb/seq: 3   name: pdbs/split_chain/6S36_A.pdb
##    PDB has ALT records, taking A only, rm.alt=TRUE
## pdb/seq: 4   name: pdbs/split_chain/6RZE_A.pdb
##    PDB has ALT records, taking A only, rm.alt=TRUE
## pdb/seq: 5   name: pdbs/split_chain/4X8H_A.pdb
## pdb/seq: 6   name: pdbs/split_chain/3HPR_A.pdb
##    PDB has ALT records, taking A only, rm.alt=TRUE
## pdb/seq: 7   name: pdbs/split_chain/1E4V_A.pdb
## pdb/seq: 8   name: pdbs/split_chain/5EJE_A.pdb
##    PDB has ALT records, taking A only, rm.alt=TRUE
## pdb/seq: 9   name: pdbs/split_chain/1E4Y_A.pdb
## pdb/seq: 10   name: pdbs/split_chain/3X2S_A.pdb
## pdb/seq: 11   name: pdbs/split_chain/6HAP_A.pdb
## pdb/seq: 12   name: pdbs/split_chain/6HAM_A.pdb
##    PDB has ALT records, taking A only, rm.alt=TRUE
## pdb/seq: 13   name: pdbs/split_chain/4K46_A.pdb
##    PDB has ALT records, taking A only, rm.alt=TRUE
## pdb/seq: 14   name: pdbs/split_chain/4NP6_A.pdb
## pdb/seq: 15   name: pdbs/split_chain/3GMT_A.pdb
## pdb/seq: 16   name: pdbs/split_chain/4PZL_A.pdb
```

```
#vector containing PDB codes for figure axis
ids <- basename.pdb(pdbs$id)

#draw schematic alignment
plot(pdbs, labels = ids)
```



Viewing the Superposed Structures [OPTIONAL]

```
#install.packages("devtools")
#library(devtools)
#install_bitbucket("Grantlab/bio3d-view")
#install.packages("rgl")

library(bio3d.view)
library(rgl)

view.pdbs(pdbs)
```

[viewer works! :)]

Annotate PDB Structures

The functions below help us annotate the PDB structures so we can associate each structure with its source species.

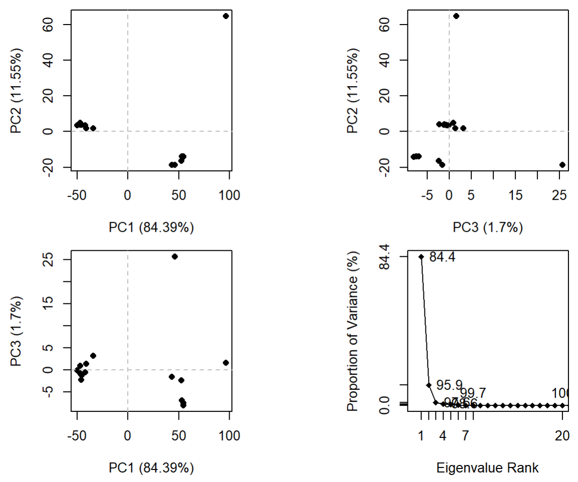
```
anno <- pdb.annotate(c("2mh3_A", "4f3l"), anno.terms = c("structureId", "experimentalTechnique", "resolution","pfam", "sourc
e", "citation"))
anno
```

```
##        structureId experimentalTechnique resolution
## 2MH3_A        2MH3                   NMR         NA
## 4F3L_B        4F3L                 X-ray      2.268
## 4F3L_A        4F3L                 X-ray      2.268
##                                             pfam       source
## 2MH3_A Helix-loop-helix DNA-binding domain (HLH) Homo sapiens
## 4F3L_B                       PAS domain (PAS_11) Mus musculus
## 4F3L_A                       PAS domain (PAS_11) Mus musculus
##                                   citation
## 2MH3_A Popovic, M., et al. Proteins (2014)
## 4F3L_B    Huang, N., et al. Science (2012)
## 4F3L_A    Huang, N., et al. Science (2012)
```

Principal Component Analysis

We can use PCA on the identified PDBs in order to determine any significant structural variations.

```
pc.xray <- pca(pdbs)
plot(pc.xray)
```



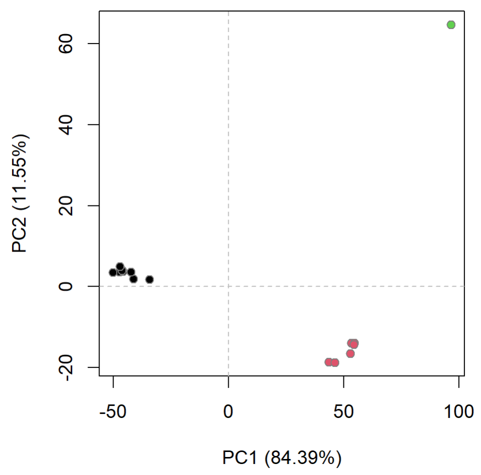
We can calculate the pairwise RMSD values of the structures, which can help with clutering analysis.

```
#calculate RMSD
rd <- rmsd(pdbs)
```

```
## Warning in rmsd(pdbs): No indices provided, using the 204 non NA positions
```

```
#structure-based clustering
hc.rd <- hclust(dist(rd))
grps.rd <- cutree(hc.rd, k = 3)

plot(pc.xray, 1:2, col="grey50", bg = grps.rd, pch = 21, cex = 1)
```



Normal Mode Analysis [OPTIONAL]

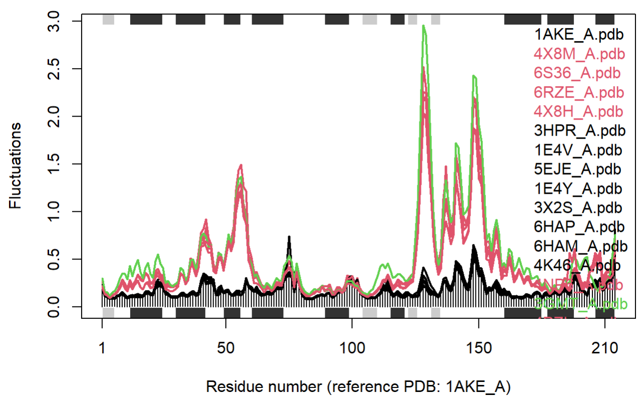
Normal Mode Analysis on PDBs can help with characterizing the profiles of related protein structures.

```
modes <- nma(pdbs)
```

```
##
## Details of Scheduled Calculation:
##   ... 16 input structures
##   ... storing 606 eigenvectors for each structure
##   ... dimension of x$U.subspace: ( 612x606x16 )
##   ... coordinate superposition prior to NM calculation
##   ... aligned eigenvectors (gap containing positions removed)
##   ... estimated memory usage of final 'eNMA' object: 45.4 Mb
##
##
  |
  |                                                                      |   0%
  |
  |====                                                                  |   6%
  |
  |=========                                                             |  12%
  |
  |=============                                                         |  19%
  |
  |==================                                                    |  25%
  |
  |======================                                                |  31%
  |
  |==========================                                            |  38%
  |
  |===============================                                       |  44%
  |
  |===================================                                   |  50%
  |
  |=======================================                               |  56%
  |
  |============================================                          |  62%
  |
  |================================================                      |  69%
  |
  |=====================================================                 |  75%
  |
  |=========================================================             |  81%
  |
  |=============================================================         |  88%
  |
  |==================================================================    |  94%
  |
  |======================================================================| 100%
```

```
plot(modes, pdbs, col = grps.rd)
```

```
## Extracting SSE from pdbs$sse attribute
```



Q14. What do you note about this plot? Are the black and colored lines similar or different? Where do you think they differ most and why?

The black lines and colored lines differ at certain regions, particularly from around residue numbers 30-70 and residue numbers 125-175. These differences may be due to those residues being associated with ligand binding sites for the structures represented by the colored lines.