Structural Bioinformatics (Pt. 1)

Matt Hashimoto

11/14/2021

PDB Statistics

We can examine a .csv file downloaded from the PDB to determine statistics about the protein data present in the database. First, let's create a matrix with the data:

```
# Store PDB summary data in a matrix
pdbSummary <- read.csv2("Data Export Summary.csv", sep = ",")
pdbSummary
```

```
##            Molecular.Type  X.ray   NMR   EM Multiple.methods Neutron Other
## 1          Protein (only) 142538 11810 6063              177      70    32
## 2 Protein/Oligosaccharide   8442    31  996                5       0     0
## 3              Protein/NA   7505   274 2008                3       0     0
## 4     Nucleic acid (only)   2368  1382   60                8       2     1
## 5                   Other    149    31    3                0       0     0
## 6  Oligosaccharide (only)     11     6    0                1       0     4
##    Total
## 1 160690
## 2   9474
## 3   9790
## 4   3821
## 5    183
## 6     22
```

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

To find this percentage, we can take the sums of the X.ray and EM columns and divide it by the total sum:

```
# Sum of X.ray and EM columns divided by total, times 100 to represent percent
sum(pdbSummary$X.ray, pdbSummary$EM) / sum(pdbSummary$Total) * 100
```

```
## [1] 92.47907
```

Thus, the percentage of structures in the PDB solved by X-Ray and Electron Microscopy is 92.48%.

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

To find this percentage, we can take the sum of the Protein (only) row and divide it by the total sum:

```
# Sum of Protein (only) total divided by total
sum(pdbSummary$Total[1]) / sum(pdbSummary$Total)
```

```
## [1] 0.8734102
```

Thus, the proportion of structures in the PDB that are protein is 0.8734.

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?"

Viewing the PDB query results for "HIV" and narrowing the search results by ensuring the macromolecule name includes "protease" yields 819 hits.

Getting to Know VMD

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

The bead display method displays residues/sidechains as a single sphere to simplify things. For example, if we were to select "bead" display for the MK1 substrate, it would appear as one single large sphere.

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 water molecule in the binding site appears to be on residue 308.

Sequence Viewer Extension

Q6 "As you have hopefully observed HIV protease is a homodimer (i.e. it is composed of two identical chains). With the aid of the graphic display and the sequence viewer extension can you identify secondary structure elements that are likely to only form in the dimer rather than the monomer?"

Yes, there is a pair of parallel beta sheets that cannot exist outside of the dimer form. These are present in both chains in the high 90s (residue number). In monomer form, these beta sheets exist alone, which is somewhat unstable. However, in dimer form, they are parallel, stabilizing one another.

Introduction to Bio3D in R

First, let's load the Bio3D library into our session:

```
# Load the Bio3D library
library(bio3d)
```

Reading PDB File Data into R

We can read a file from the PDB database directly into R:

```
# Load the PDB file associated with 1HSG
pdb <- read.pdb("1hsg")
```

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

Next we can examine the contents of this file:

```
# View loaded 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
```

Q7 "How many amino acid residues are there in this PDB object?"

There are 198 protein residues and 128 non-protein residues, for a total of 326 residues.

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

MK1 is a non-protein residue.

Q9 "How many protein chains are in this structure?"

This structure has 2 chains.

We can take a quick look at the attributes of the PDB object:

```
# Check attributes
attributes(pdb)
```

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

```
# Access the atom attribute
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>
```

Setup

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

The msa package is only found on BioConductor and not CRAN.

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

The bio3d-view package is not found on BioConductor or CRAN.

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

True

Search and Retrieve ADK Structures

First, let's load in the Adenylate kinase sequence for chain A:

```
# Load the Bio3D library and obtain sequence for chain A of 1AKE
library(bio3d)
aa <- get.seq("1ake_A")
```

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

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

```
# Check sequence
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?"

This sequence is 214 amino acids long.

The next step is to perform a BLAST search using our sequence:

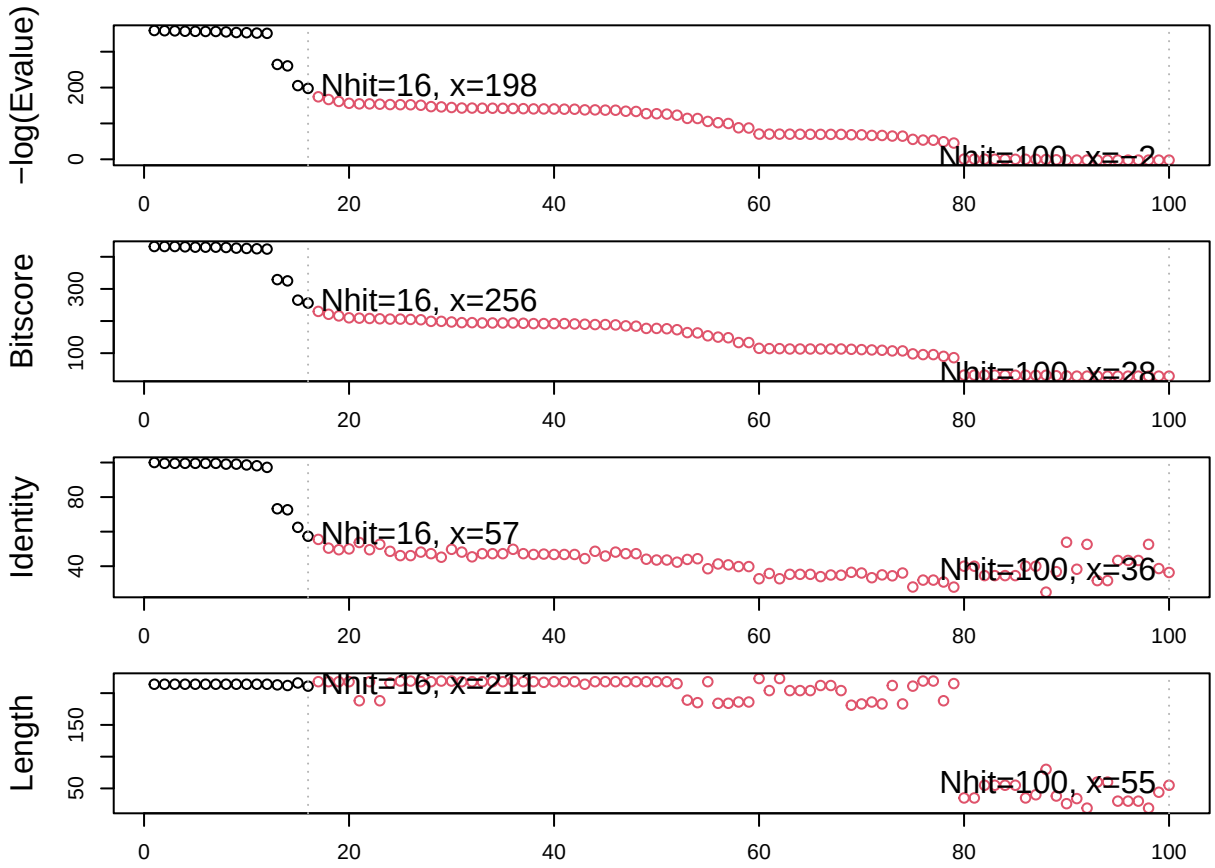
```
# BLAST or HMMER search
b <- blast.pdb(aa)
```

```
##  Searching ... please wait (updates every 5 seconds) RID = T4FK2EKK013
##  ..............................
##  Reporting 100 hits
```

Let's narrow down the search a bit to only include E. coli results:

```
# Plot a summary of search results
hits <- plot(b)
```

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



We will thus focus only on the black results.

```
# List out some 'top hits'
head(hits$pdb.id)
```

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

Now that we have our top hits, we can retrieve the associated PDB files:

```
# 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.gz exists. Skipping download
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

##   |                                                                              |
```

Align and Superpose Structures

We can now begin to align the PDB structures we found and fit them:

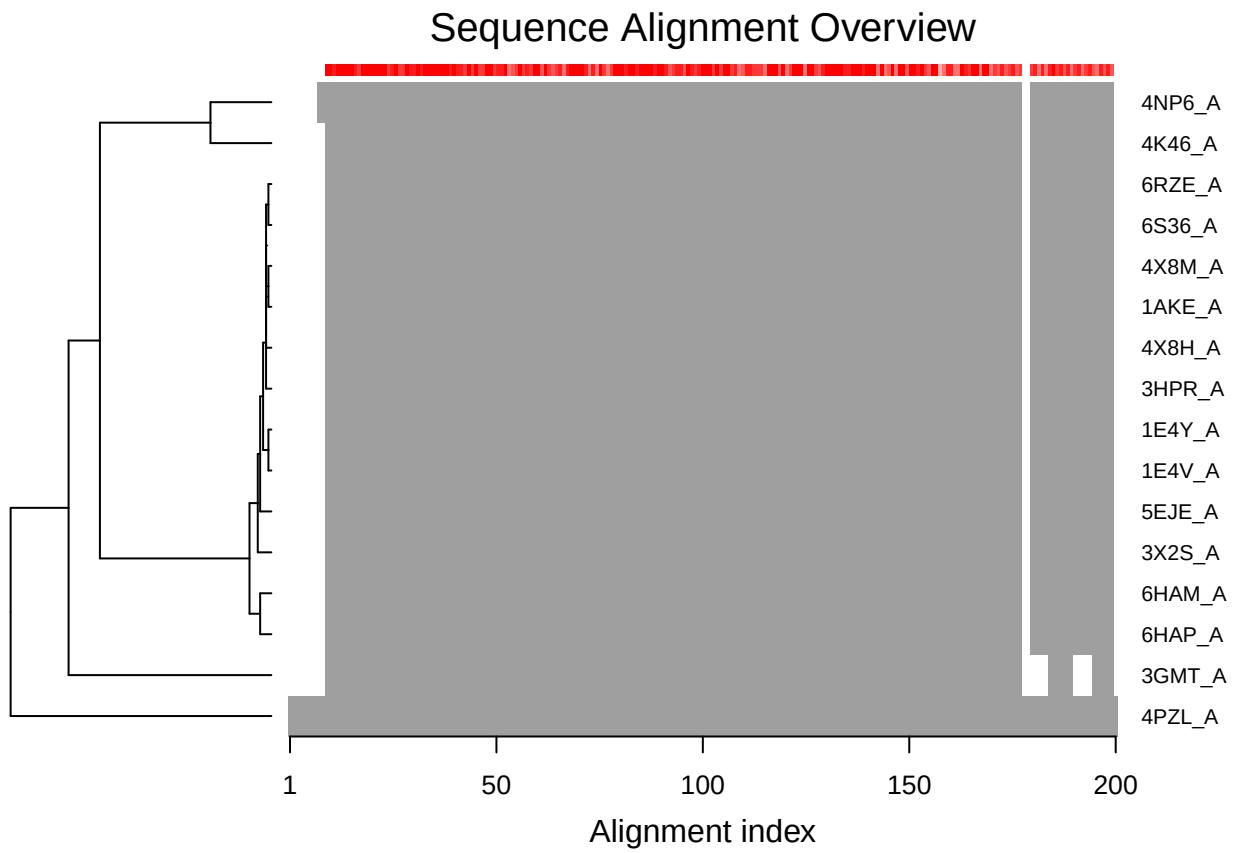
```
# Align related PDBs
pdbs <- pdbaln(files, fit = TRUE, exefile="msa")
```

```
## 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
```

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

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



Viewing Our Superposed Structures

We can now view our superposed structures in 3D space:

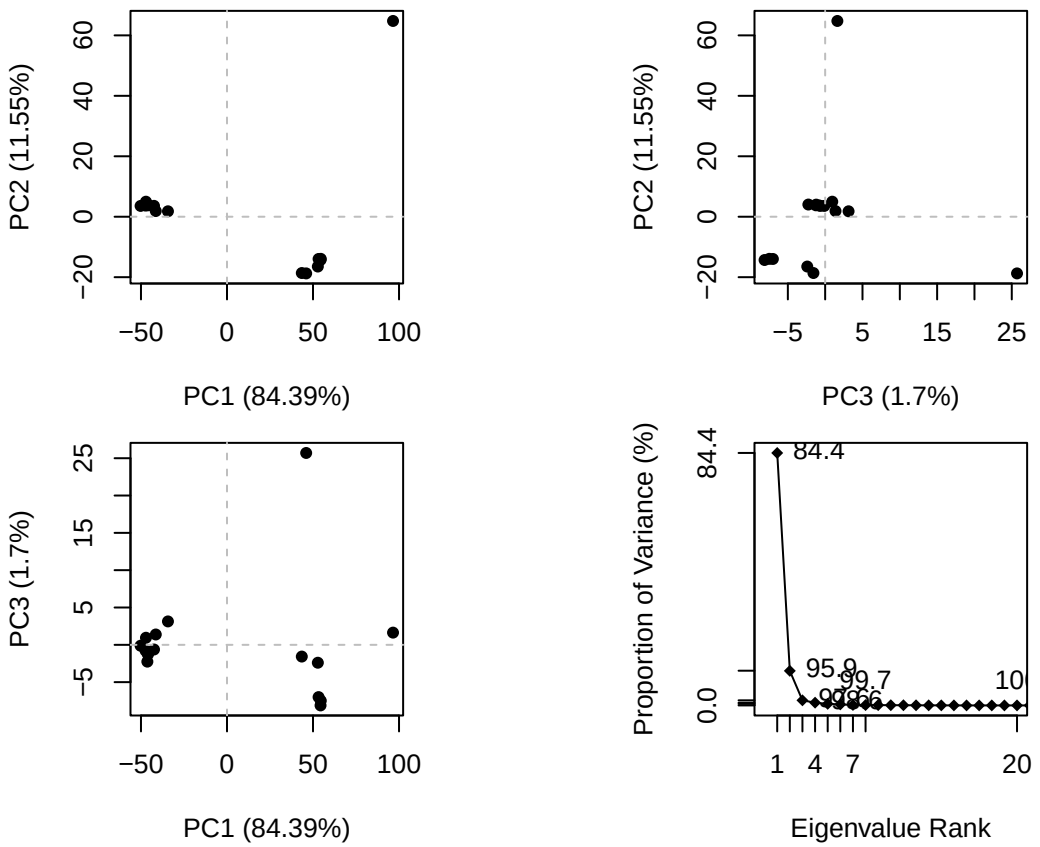
```
# Load necessary libraries
library(bio3d.view)
library(rgl)

# Create visualization
#view.pdbs(pdbs)
```

Principal Component Analysis

We can perform PCA on the group of PDB structures we obtained earlier:

```
# Perform PCA
pc.xray <- pca(pdbs)
plot(pc.xray)
```



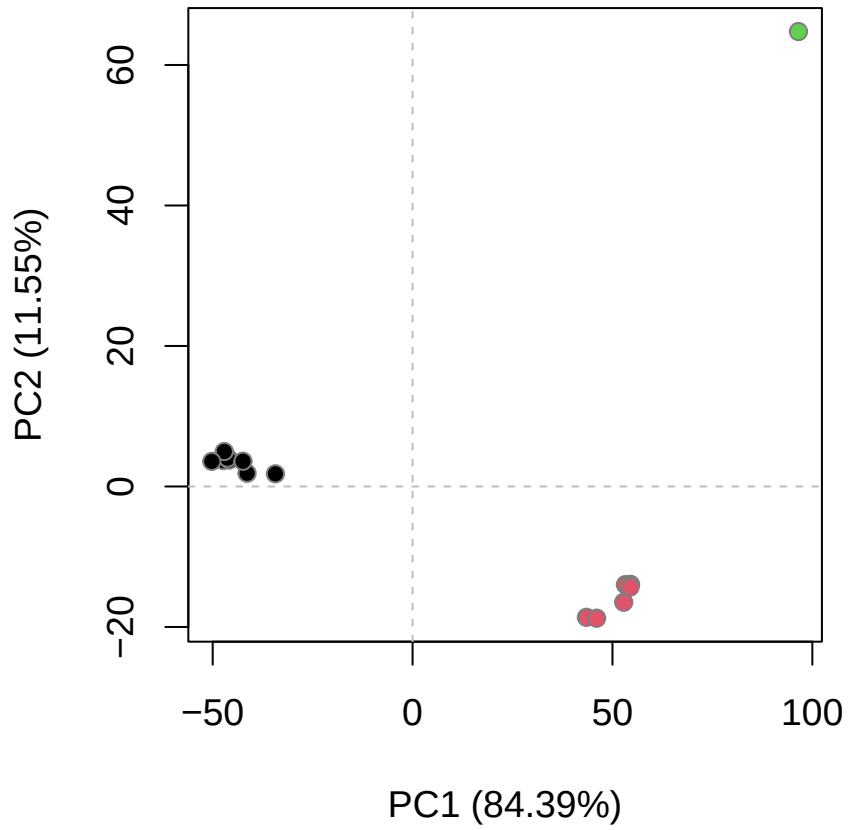
Next, calculating RMSD values can assist us in clustering analysis:

```
# 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

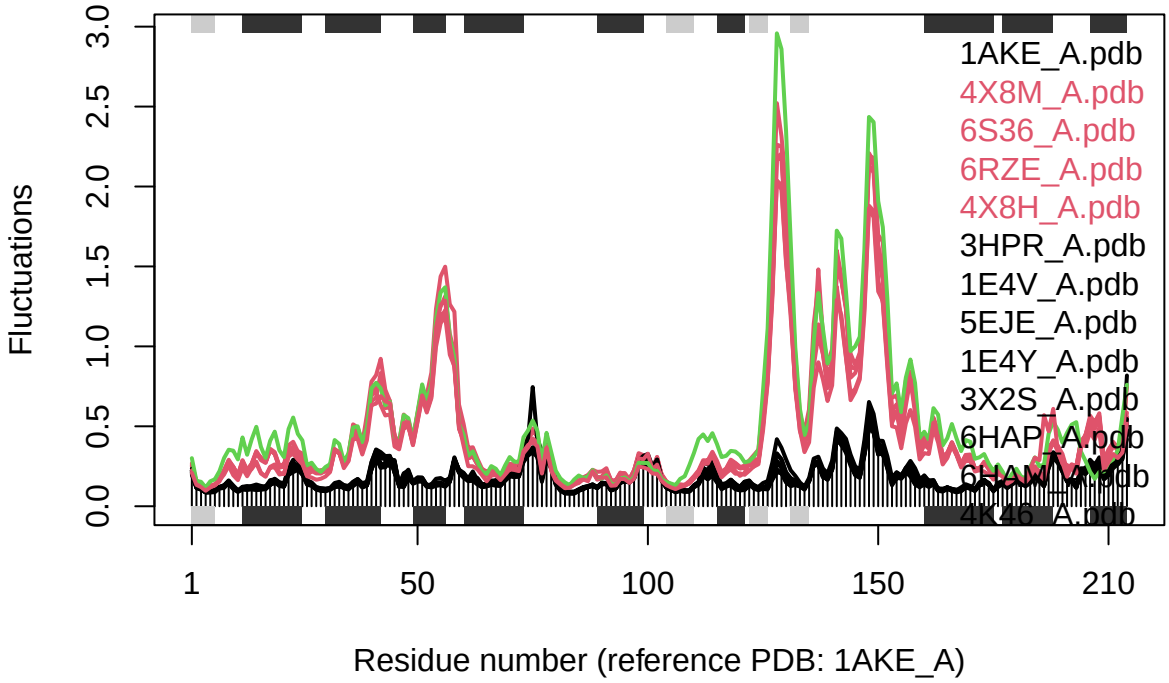
Lastly, we can analyze either a single PDB object or a group of PDB objects using NMA, allowing us to investigate flexibility profiles of protein structures:

```
# NMA of all 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
##
##    |                                                                      |
```

```
# Graphical analysis
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?"

There are two distinct groups (black and colored) that are similar to one another, but different from all those in the other gorup. The black and colored lines are decently similar in some areas and quite different in others. They likely differ near their binding sites, as conformational changes can vastly impact flexibility at and near those regions.