Structural Bioinformatics (Pt. 1)

Joshua Cheung

02/13/2022

1. Introduction ot the RCSB Protein Data Bank (PDB)

PDB statistics

We download the CSV files from the PDB site and access it as follows.

```
# Here we save the input data file into our Project directory.
PDB.data <- "Data Export Summary.csv"
# We assign the result of the above code to an object called PDB.df.
PDB.df <- read.csv(PDB.data, row.names=1)
PDB.df
```

```
##                           X.ray   NMR   EM Multiple.methods Neutron Other  Total
## Protein (only)           144301 11877 6676              182      70    32 163138
## Protein/Oligosaccharide    8528    31 1116                5       0     0   9680
## Protein/NA                 7617   274 2153                3       0     0  10047
## Nucleic acid (only)        2393  1398   61                8       2     1   3863
## 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.**

```
# The percentage of structures in the PDB solved by X-ray is:
round((sum(PDB.df$X.ray)/sum(PDB.df$Total))*100, digits=2)
```

```
## [1] 87.2
```

```
# The percentage of structures in the PDB solved by EM is:
round((sum(PDB.df$EM)/sum(PDB.df$Total))*100, digits=2)
```

```
## [1] 5.35
```

Thus, the percentage of structures in the PDB solved by X-Ray is 87.2% and the percentage of structures solved in the PDB solved by Electron Microscopy is 5.35%.

> **Q2. What proportion of structures in the PDB are protein?**

```
# The proportion of structures in the PDB that are protein is:
round((sum(PDB.df[1,]) - PDB.df[1,"Total"])/sum(PDB.df$Total), digits=3)
```

```
## [1] 0.873
```

Thus, the proportion of structures in the PDB that are protein is 0.873.

> **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?**

After typing HIV into the PDV website search box on the home page and filtering the results by HIV type 1, there were still approximately 837 results according to the PDB website. However, it is difficult to say for certain whether all these results are relvant to what we are looking for. Thus, we conclude that the answer to how many HIV-1 protease structures is not easy to determine based solely from a text search with keywords like "HIV" or "HIV-1 protease." Using a sequence search would get a much more reliable set of results.

The PDB format

We download the PDB file for the HIV-1 protease structure with the PDB identify 1HSG. We can examine the contents of the downloaded file using the Terminal tab from within RStudio with the following command.

```
# less ~/Downloads/1hsg.pdb  ## (use 'q' to quit)
# Un-comment the above code and enter it in the Terminal tab to examine the
# contents of the downloaded file.
```

2. Visualizing the HIV-1 protease structure

Using atom selections

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

If we want to select and display all water molecules as red spheres we can use the selection text "resname HOH" and use the VDW drawing method and name coloring method. In this case we see "one atom" per water molecule in this structure because the VDW method draws the atoms as spheres with the radius for each atom set to the van der Waals radius. In water, oxygen is much larger than hydrogen, so oxygen exhibits greater van der Waals forces as it has a larger electron cloud surface area. By comparison, hydrogen exhibits much smaller van der Waals forces due to its much smaller electron cloud surface area. Thus the van der Waal sphere for hydrogen is much smaller and is entirely dwarfed by oxygen's vaan der Waal sphere hence we only see one atom (oxygen) per 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)?**

Yes, we can identify this water molecule. It's residue number is HOH308:0.

> **OPTIONAL: Generate and save a figure clearly showing the two distinct chains of HIV-protease along with the ligand. You might also consider showing the catalytic residues ASP 25 in each chain (we recommend Licorice for these side-chains). Upload this figure to Piazza for some extra credit.**

The figure showing distinct chains of the HIV-protease as well as the ligand and the catalytic residues has been posted to Piazza.

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

Indinavir could enter the binding site by being analog or a derivative from the same class of molecule as the proteases natural ligand allowing indinavir to competitively inhibit the protease ligand. Similarly, a larger ligand or substrate could enter the binding site, by having the portion of the larger ligand of substrate that interacts with the binding site, be an analog of the proteases natural ligand. In the case of larger ligands and substrate it is important to note that while the rest of the larger molecule that does not interact with the binding site, must not block any binding site interactions and must not cause steric hindrance.

Sequence viewer extension [OPTIONAL]

> **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?**

While looking through the sequence viewer, we see that there is a single instance of an amino acid residue involved in a secondary structure called an isolated bridge (labelled "B"). This indicates there is an isolated hydrogen bond connecting elsewhere, likely the other dimer. Thus it is likely that the isolated bridge secondary structure is only forms in the dimer rather than the monomer.

3. Introduction to Bio3D in R

We load the Bio3D package by using the following code:

```r
# install.packages("bio3d") ## Un-comment to install if necessary.
library(bio3d)
```

Reading PDB file data into R

We use the read.pdb() function to read a single PDB file with Bio3D.

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

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

We get a quick summary of the contents of the pdb object we just created using the following code:

```
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 amino acid resiudes in this pdb object.

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

One of the non-protein residues is MK1.

Q9. How many protein chains are in this structure?

There are 2 protein chains in this structure.

We now note that the attribute of this pdb object are listed in the last couple lines. To find the attributes of any such pdb object we can use the following code.

```
attributes(pdb)
```

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

Additionally, we note that in order to access these individual attributes we can use the dollar-attribute name convention. For instance is we wished to access the atom attribute, we can use pdb$atom.

```
# We preview the first 6 rows only as pdb is a large object.
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>
```

4. Comparative structure analysis of adenylate kinase

Setup

We begin this section by first installing the packages necessary.

```
# We install the following packages in the R console and not this document.
## Un-comment any of the following lines to install if necessary

# install.packages("bio3d")
# install.packages("ggplot2")
# 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 in 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 retrive ADK structures

We use the function get.seq() in order to fetch the query sequence for chain A of the PDB ID 1AKE. We use this as an input to the function plst.pdb().

```
library(bio3d)
aa <- get.seq("1ake_A")
```

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

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

We now examine the contents of “aa”.

```
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 this sequence.

We can now use this sequence to query to BLAST search the PDB in order to find similar sequences and stuctures.

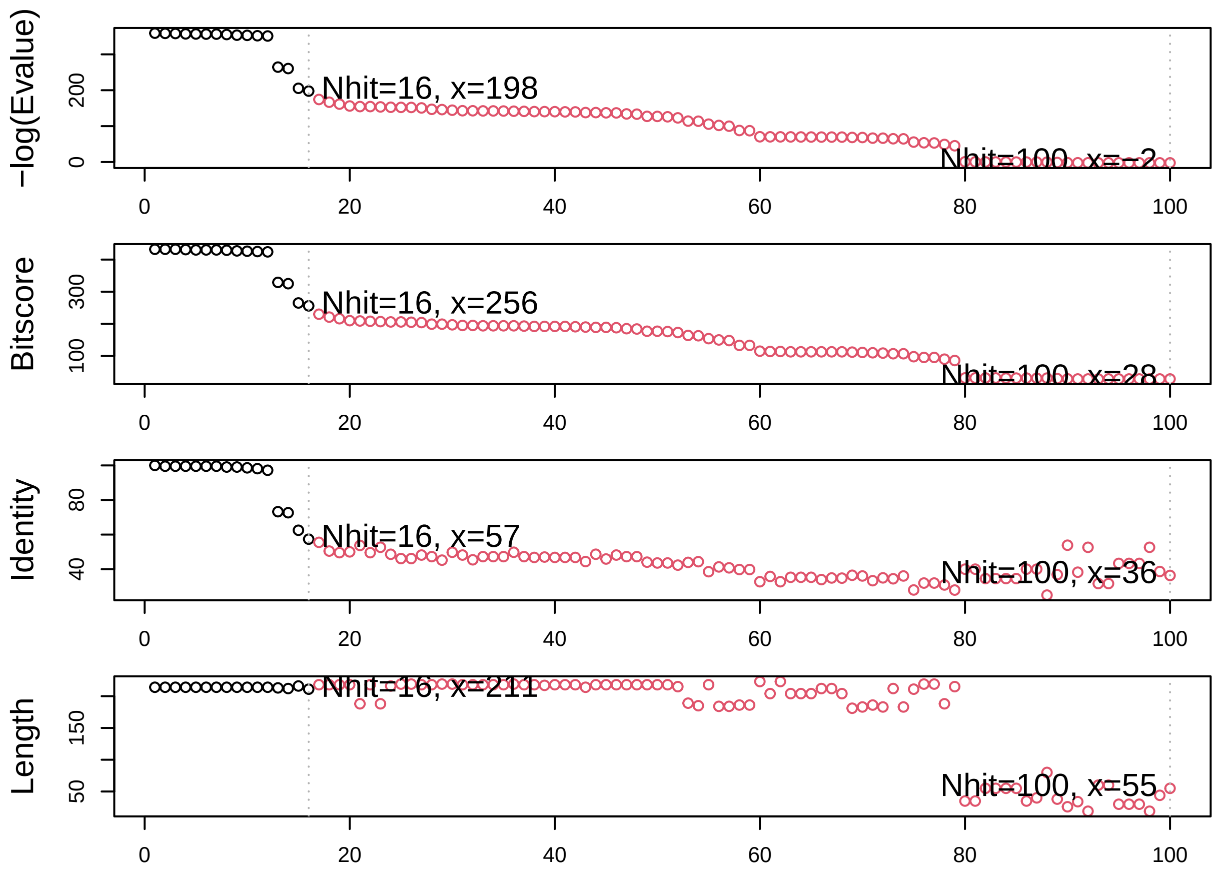
```
# We perform a blast or hmmer search.
b <- blast.pdb(aa)
```

```
##  Searching ... please wait (updates every 5 seconds) RID = 2D3ECWJP013
##  .
##  Reporting 100 hits
```

We use the function plot.blast() to facilitate the visualization and filtering of the blast results.

```
# We 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
```



Next we proceed with only the top scoring hits (in black).

```
# We list out some of the "top hits".
# We examine the first 6 elements of hits$pdb.id.
head(hits$pdb.id)
```

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

We can now use function get.pdb() and pdbslit() to fetch and parse the identified structures.

```r
# We now download the related pdb files.
files <- get.pdb(hits$pdb.id, path="pdbs", split=TRUE, gzip=TRUE)
```

```
##   |                                                                              |
```

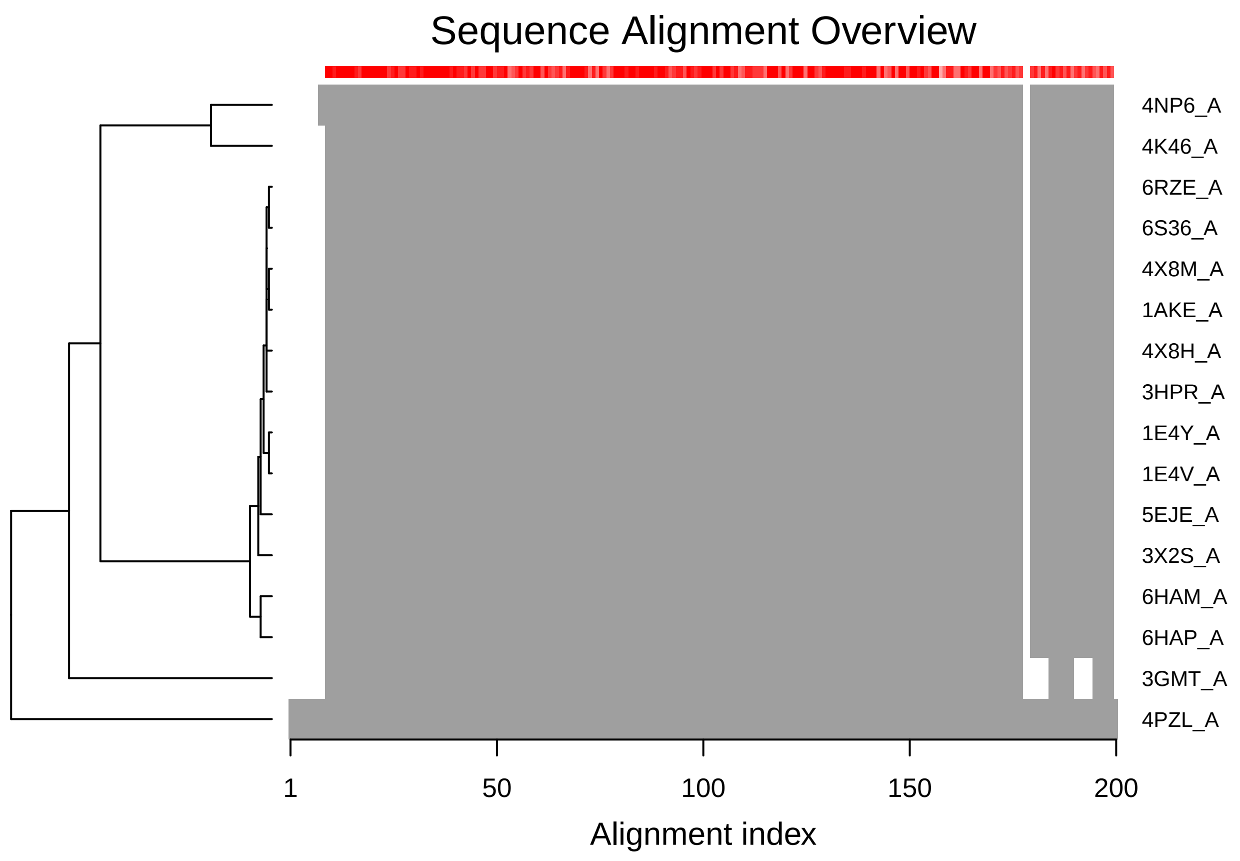
Align and superpose structures

We will now use the pdbaln() function to align and also optionally superimpose the identified pdb structures.

```r
# We first align the 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
```

```
# We create a vector containing PDB codes for the figure axis.
ids <- basename.pdb(pdbs$id)
# We now draw the  alignment.
plot(pdbs, labels=ids)
```



Thus this figure is a schematic representation of the alignment. The grey regions depict aligned residues, while white depict gap regions. The red bar at the top depict sequence conservation.

Optional: Viewing our superposed structures

We view our superposed results with the new bio3d.view view() function:

```
library(bio3d.view)
library(rgl)
view.pdbs(pdbs)
```

Note that the structure will not be shown on the PDF version of this document. Nonetheless, we include the code so that the reader can see what was done.

Optional: Annotate collected PDB structures.

We use the pdb.annotate() function as a convenient way of annotating the PDB files we have collected. We use the function to annotate each structure to its source species.

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

```
## [1] "Homo sapiens" "Mus musculus"
```

We can now view all available annotation data as follows.

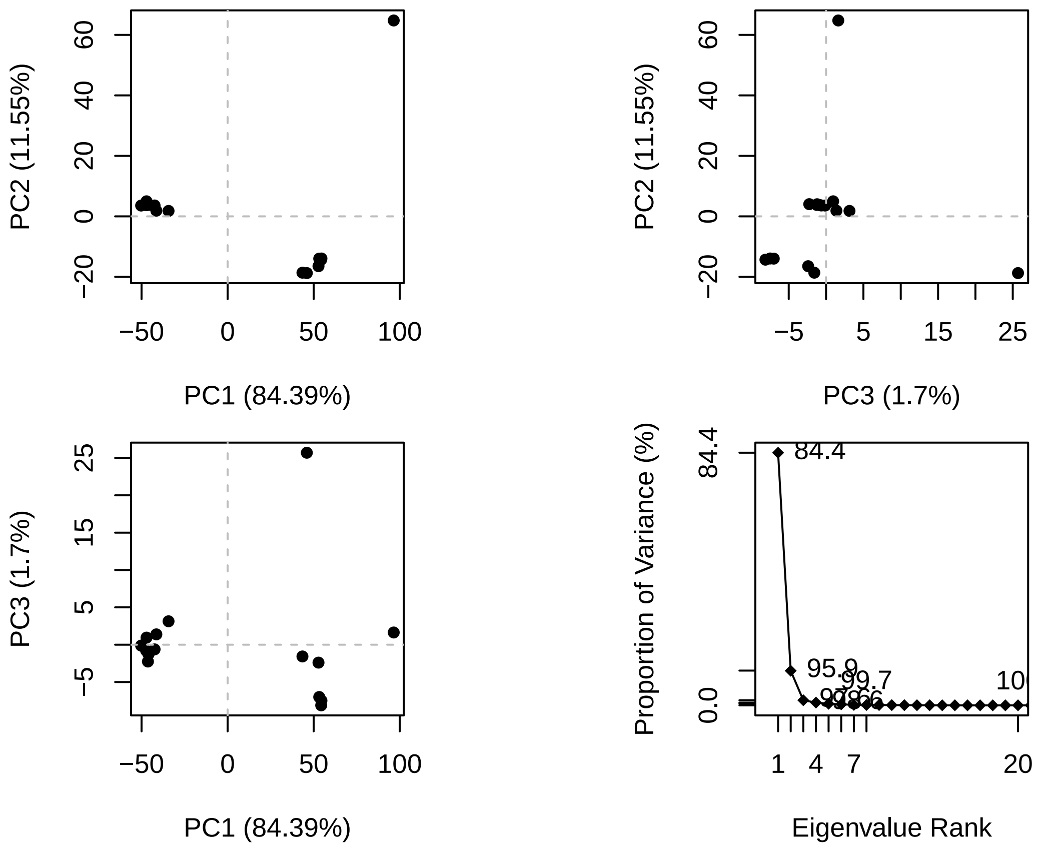
```
anno
```

```
##        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 fold (PAS) Mus musculus
## 4F3L_A                           PAS fold (PAS) 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 perform PCA on the structural ensemble stored in the pdbs object with the function pca() or pca.xyz(). We use pca() below.

```
# We now perform PCA.
pc.xray <- pca(pdbs)
plot(pc.xray)
```

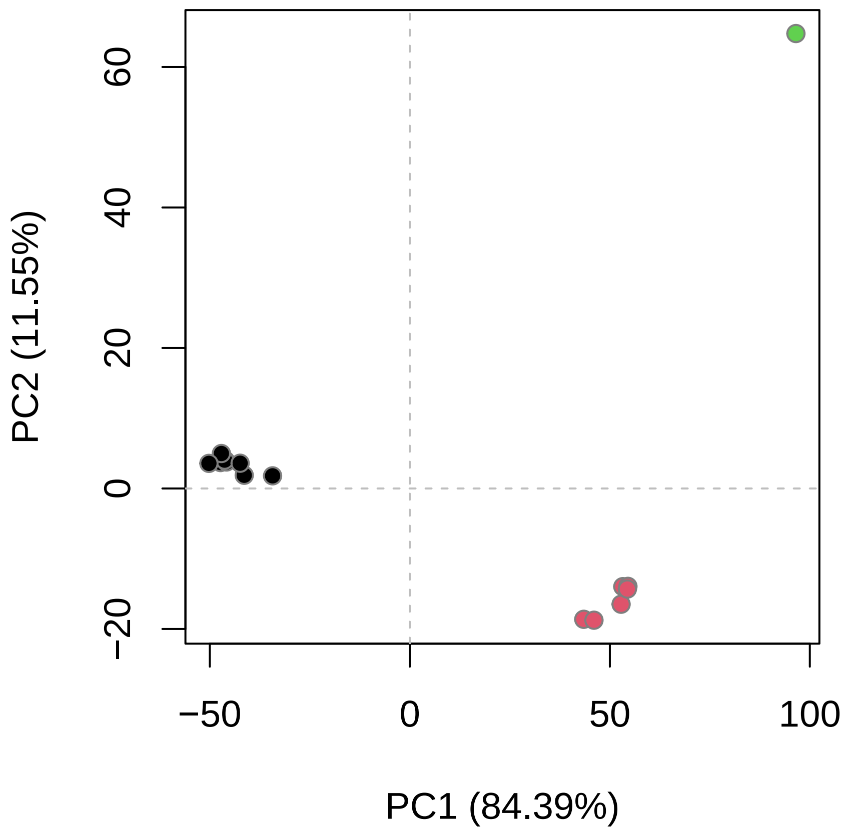


We can then use the rmsd() function to calculate all pairwise RMSD values of the structural ensemble, facilitating clustering analysis based on the pairwise strucural deviation.

```
# We calculate the RMSD.
rd <- rmsd(pdbs)
```

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

```
# Now we perform structural-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)
```



The plot above is obtained by projecting the individual structures onto two selected PCs. These projections display the inter-conformer relationship in terms of the conformational differences described by the selected PCs. This is called a conformer plot and is a low-dimensional representation of the conformational variability within the ensemble of PDB structures.

5. Optional further visualization

We visualize the major structural variations in the ensemble with the function mktrj(). This generates a trajectory PDB file by interpolating along a given PC.

```
# We visualize first principal component.
pc1 <- mktrj(pc.xray, pc=1, file="pc_1.pdb")
```

We can not only view our results with VMD but also with the bio3d.view view() function as follows. Note that once again the three dimensional view of our results will not be available to view on the PDF function. Nevertheless we include the code used for the reader to see.

```
view.xyz(pc1)
```

```
## Potential all C-alpha atom structure(s) detected: Using calpha.connectivity()
```

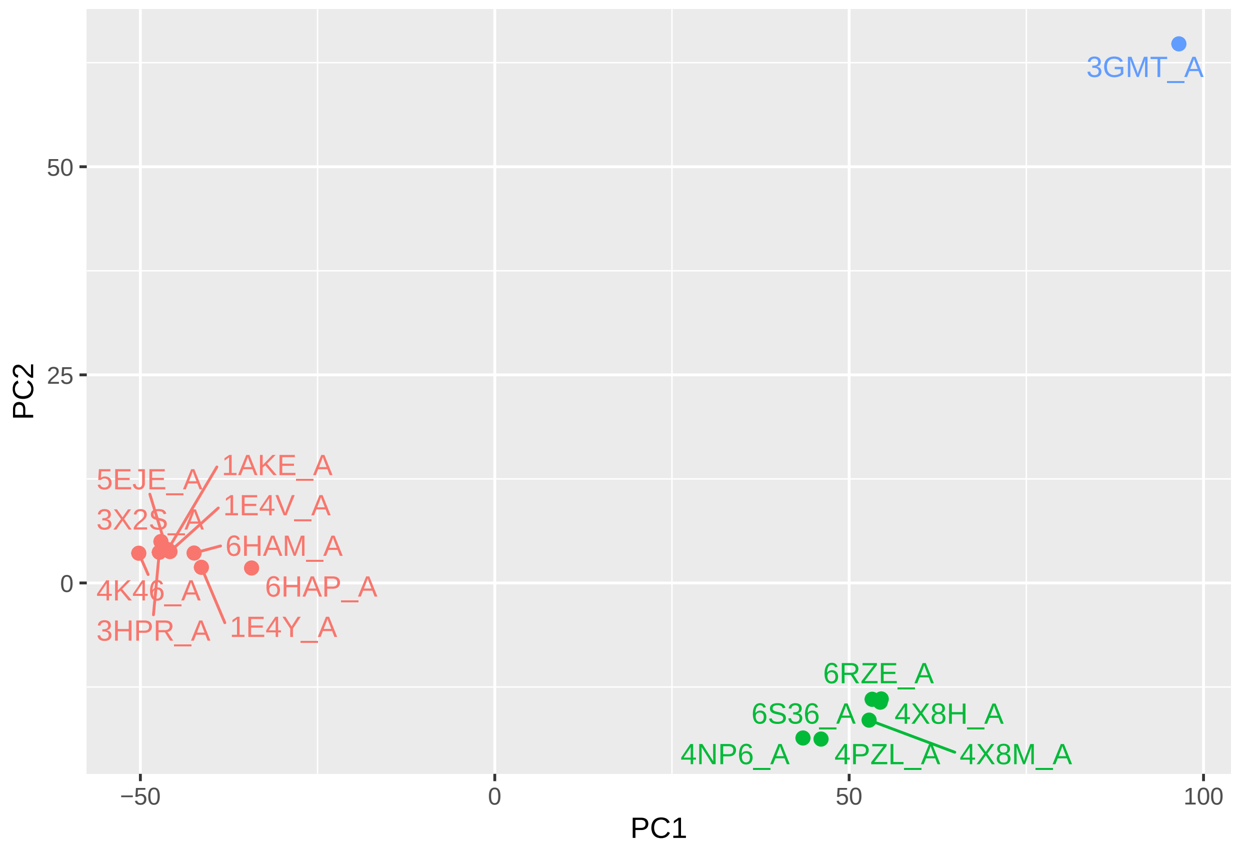
We now edit the code to set the color to highlight the most variable regions as follows.

```
view.xyz(pc1, col=vec2color( rmsf(pc1) ))
```

```
## Potential all C-alpha atom structure(s) detected: Using calpha.connectivity()
```

Finally, we can also plot our main PCA results using ggplot2.

```
library(ggplot2)
library(ggrepel)
# We define a data frame.
df <- data.frame(PC1=pc.xray$z[,1],
                 PC2=pc.xray$z[,2],
                 col=as.factor(grps.rd),
                 ids=ids)
# We can now plot the results with ggplot2.
p <- ggplot(df) +
  aes(PC1, PC2, col=col, label=ids) +
  geom_point(size=2) +
  geom_text_repel(max.overlaps = 20) +
  theme(legend.position = "none")
p
```



6. Normal mode analysis

We now use the function nma() for normal mode analysis. This function can be used for both single structures and complete structure ensembles allowing characterization and comparison of flexibility profiles of related protein structures.

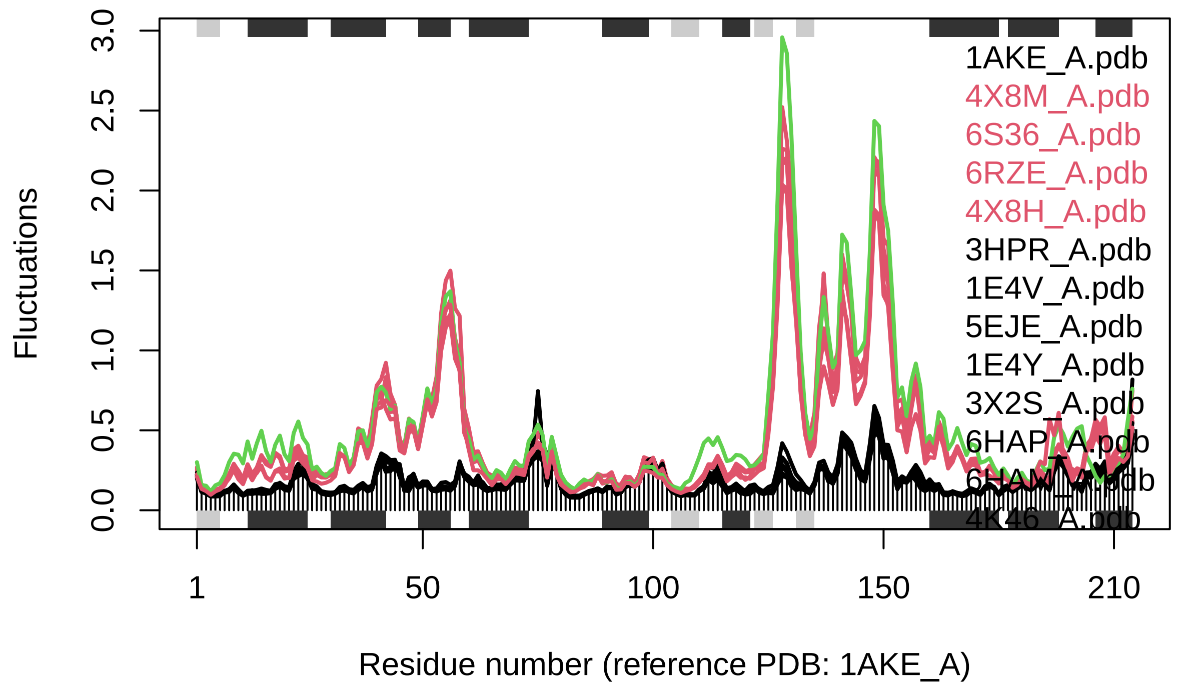
```r
# We find the NMA of all structures.
modes <- nma(pdbs)
```

```
##
## 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
##
##   |                                                                      |
```

Now we create a plot as follows:

```r
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?

In this this plot the black and colored line are clearly different at 2 specific regions. There is clear divergence around the the 25-60 residue number range and from the 125-175 residue number range. Aside from these two regions, the black and colored lines appear to be close to each other. A reason for the divergence in the two specific region listed is that there are two major distinct conformational states for Adk. These two states differ by a collective low frequency displacement of two nucleotide-binding sites that display distinct flexibilities upon binding of nucleotides. Hence we see two particular areas on the plot where the black and colored lines show clear diveregence.