Class 12: Structural Bioinformatics II

Anita Wang (PID: A15567878)

11/4/2021

3. Introduction to Bio3D in R

```
library(bio3d)
```

- Reading PDB file data into R

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

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

- To get a quick summary of the contents of the pdb object you just created you can issue the command print(pdb) or simply type pdb (which is equivalent in this case):

```
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?

- 198

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

- MK1

Q9: How many protein chains are in this structure?

- 2

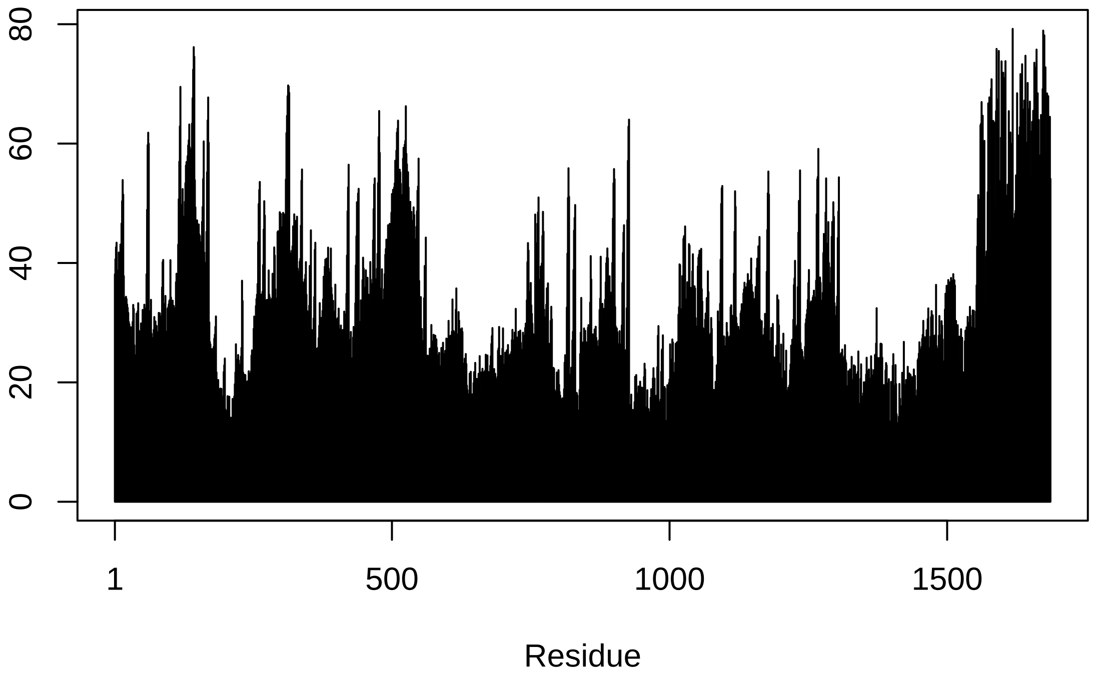
```
aa123(pdbseq(pdb))
```

```
##   [1] "PRO" "GLN" "ILE" "THR" "LEU" "TRP" "GLN" "ARG" "PRO" "LEU" "VAL" "THR"
##  [13] "ILE" "LYS" "ILE" "GLY" "GLY" "GLN" "LEU" "LYS" "GLU" "ALA" "LEU" "LEU"
##  [25] "ASP" "THR" "GLY" "ALA" "ASP" "ASP" "THR" "VAL" "LEU" "GLU" "GLU" "MET"
##  [37] "SER" "LEU" "PRO" "GLY" "ARG" "TRP" "LYS" "PRO" "LYS" "MET" "ILE" "GLY"
##  [49] "GLY" "ILE" "GLY" "GLY" "PHE" "ILE" "LYS" "VAL" "ARG" "GLN" "TYR" "ASP"
##  [61] "GLN" "ILE" "LEU" "ILE" "GLU" "ILE" "CYS" "GLY" "HIS" "LYS" "ALA" "ILE"
##  [73] "GLY" "THR" "VAL" "LEU" "VAL" "GLY" "PRO" "THR" "PRO" "VAL" "ASN" "ILE"
##  [85] "ILE" "GLY" "ARG" "ASN" "LEU" "LEU" "THR" "GLN" "ILE" "GLY" "CYS" "THR"
##  [97] "LEU" "ASN" "PHE" "PRO" "GLN" "ILE" "THR" "LEU" "TRP" "GLN" "ARG" "PRO"
## [109] "LEU" "VAL" "THR" "ILE" "LYS" "ILE" "GLY" "GLY" "GLN" "LEU" "LYS" "GLU"
## [121] "ALA" "LEU" "LEU" "ASP" "THR" "GLY" "ALA" "ASP" "ASP" "THR" "VAL" "LEU"
## [133] "GLU" "GLU" "MET" "SER" "LEU" "PRO" "GLY" "ARG" "TRP" "LYS" "PRO" "LYS"
## [145] "MET" "ILE" "GLY" "GLY" "ILE" "GLY" "GLY" "PHE" "ILE" "LYS" "VAL" "ARG"
## [157] "GLN" "TYR" "ASP" "GLN" "ILE" "LEU" "ILE" "GLU" "ILE" "CYS" "GLY" "HIS"
## [169] "LYS" "ALA" "ILE" "GLY" "THR" "VAL" "LEU" "VAL" "GLY" "PRO" "THR" "PRO"
## [181] "VAL" "ASN" "ILE" "ILE" "GLY" "ARG" "ASN" "LEU" "LEU" "THR" "GLN" "ILE"
## [193] "GLY" "CYS" "THR" "LEU" "ASN" "PHE"
```

Plot of B-factor

```
plot.bio3d(pdb$atom$b, sse=pdb)
```

```
## Warning in plotb3(...): Length of input 'sse' does not equal the length of input
## 'x'; Ignoring 'sse'
```



The ATOM records

```
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>
```

Note that the attributes (+ attr:) of this object are listed on the last couple of lines. To find the attributes of any such object you can use:

```
attributes(pdb)
```

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

To access these individual attributes we use the dollar-attribute name convention that is common with R list objects. For example, to access the atom attribute or component use pdb$atom:

```
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

- Goal of this section is to perform principal component analysis (PCA) on the complete collection of Adenylate kinase structures in the protein data-bank (PDB)

Using the bio3d package

```
library(bio3d)

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

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

```
pdb
```

```
##
##  Call:  read.pdb(file = "1hel")
##
##    Total Models#: 1
##      Total Atoms#: 1186,  XYZs#: 3558  Chains#: 1  (values: A)
##
##      Protein Atoms#: 1001  (residues/Calpha atoms#: 129)
##      Nucleic acid Atoms#: 0  (residues/phosphate atoms#: 0)
##
##      Non-protein/nucleic Atoms#: 185  (residues: 185)
##      Non-protein/nucleic resid values: [ HOH (185) ]
```

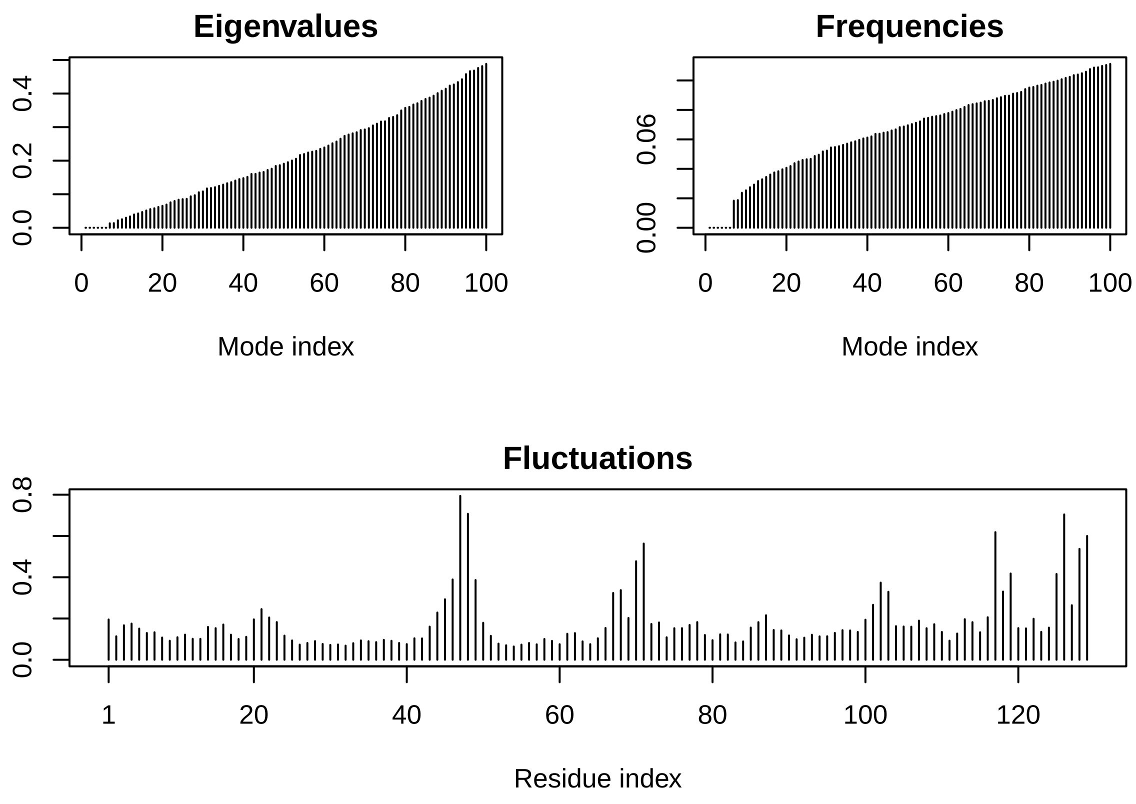
```
##
##    Protein sequence:
##       KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINS
##       RWWCNDGRTPGSRNLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWVAWRNRCKGTDV
##       QAWIRGCRL
##
## + attr: atom, xyz, seqres, helix, sheet,
##         calpha, remark, call
```

Let's use a bioiformatics mthod called NMA (Normal Mode Analysis) to predict the dynamics (flexibility) of this enzyme.

```
modes <- nma(pdb)
```

```
##  Building Hessian...     Done in 0.02 seconds.
##  Diagonalizing Hessian...    Done in 0.11 seconds.
```

```
plot(modes)
```



Now let's make a "movie" of its predicted motion. This is often called a "trajectory".

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

We can now open up this new file in VMD. Changing the coloring method to index and drawing method to tube, we can visualize the protein. Clicking the play button the in lower right hand corner of VMD Main let's the movie play and the protein move.

```
#![](nma.pdb.movie.png)
```

Rmd will not knit with the insertion of png image so I put it as a comment in a code chunk…

Overview

- Starting from only one Adk PDB identifier (PDB ID: 1AKE) we will search the entire PDB for related structures using BLAST, fetch, align and superpose the identified structures, perform PCA and finally calculate the normal modes of each individual structure in order to probe for potential differences in structural flexibility.

Setup

- First, install the necessary R-packages

```
# Install packages in the R console not your Rmd

#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?

- msa

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

- Grantlab/bio3d-view

> 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

```
library(bio3d)
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?

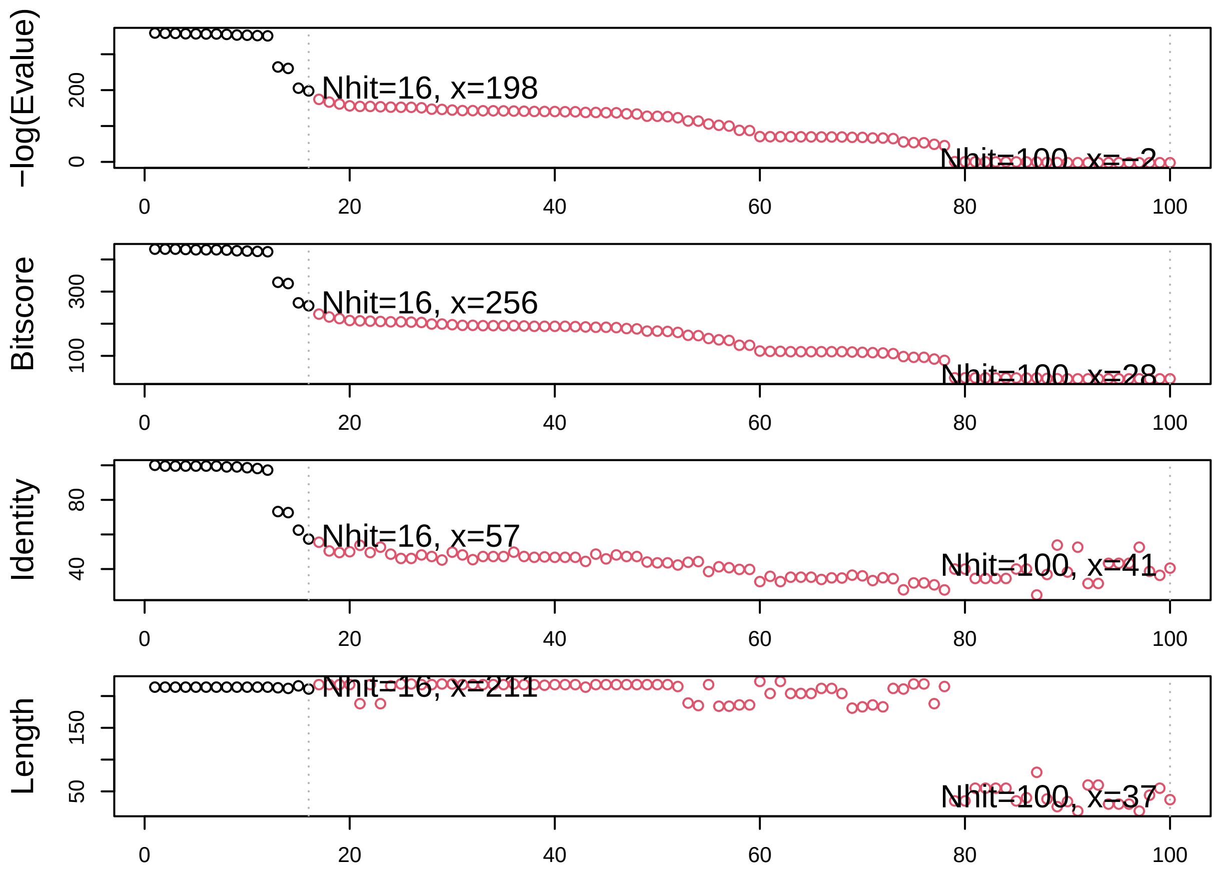
- 214

```
# Now you can run BLAST from R
blast <- blast.pdb(aa)
```

```
##  Searching ... please wait (updates every 5 seconds) RID = SGM8C90F013
##  ..................................
##  Reporting 100 hits
```

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

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



```
hits$pdb.id
```

```
##  [1] "1AKE_A" "4X8M_A" "6S36_A" "6RZE_A" "4X8H_A" "3HPR_A" "1E4V_A" "5EJE_A"
##  [9] "1E4Y_A" "3X2S_A" "6HAP_A" "6HAM_A" "4K46_A" "4NP6_A" "3GMT_A" "4PZL_A"
```

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

##   |                                                                              |
```

ˆ Takes hits files and makes a directory ("pdbs") and puts it there – like manually going and downloading all pdb files from the pdb data base and compiling it all into a single file.

What we've done: workflow –> 1AKE -> sequenced it -> blast against PDB database -> get all structures that we want to do comparative analysis on -> alignment

Now that we have all of the pdb files…

Lets **Align and superpose structures (multiple structure sequence alignment)**

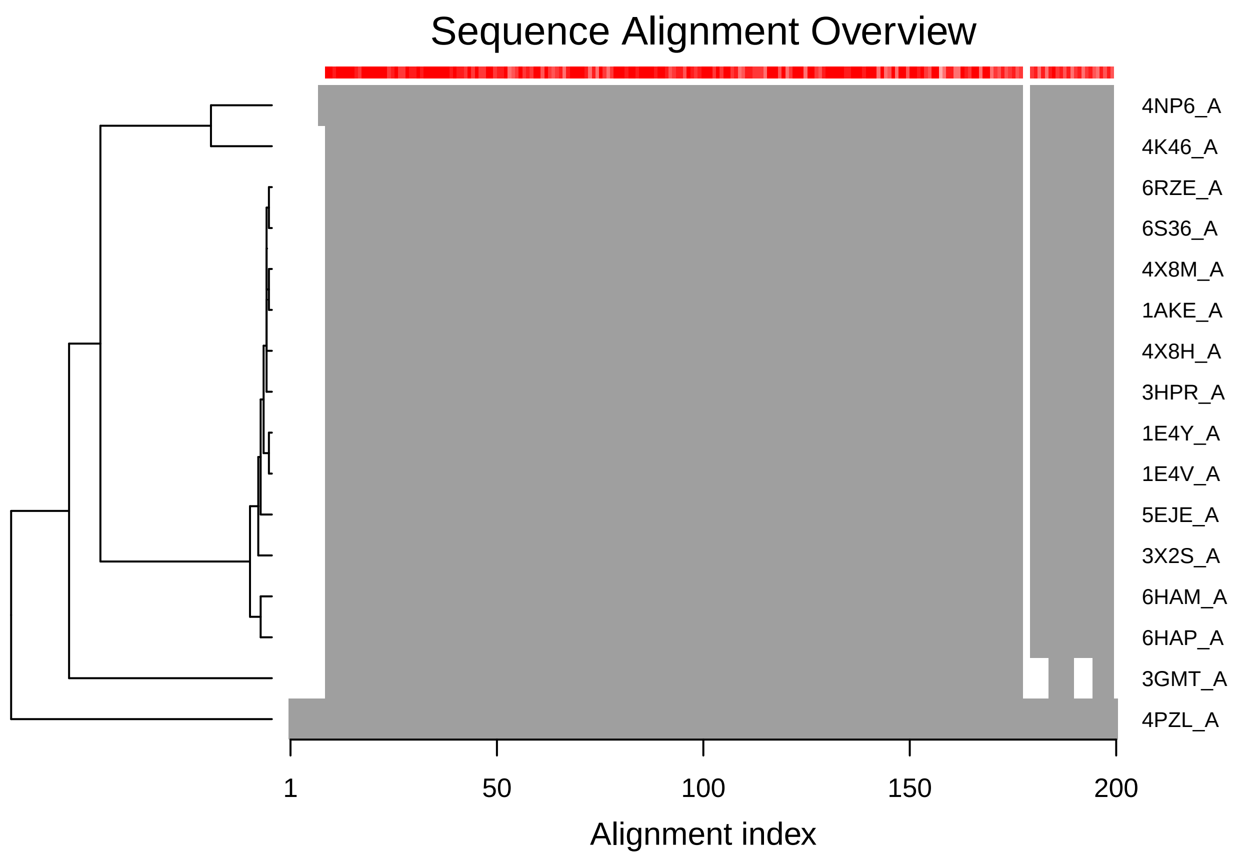
We will use the pdbaln() function to align and also optionally fit (i.e. superpose) the identified PDB structures

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

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

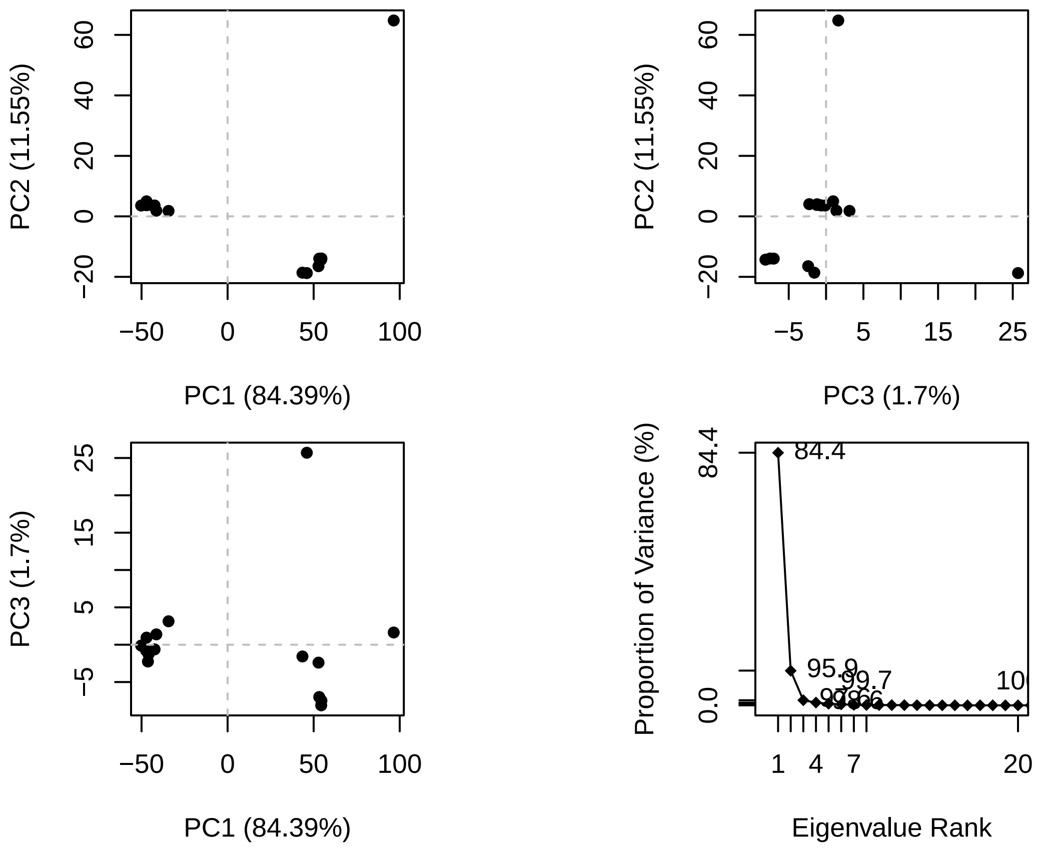


ˆA schematic representation of alignment. Grey regions depict aligned residues, while white depict gap regions. The red bar at the top depict sequence conservation.

Principal component analysis

- Function pca() provides principal component analysis (PCA) of the structure data. PCA is a statistical approach used to transform a data set down to a few important components that describe the directions where there is most variance. In terms of protein structures PCA is used to capture major structural variations within an ensemble of structures.
- PCA can be performed on the structural ensemble (stored in the pdbs object) with the function pca.xyz(), or more simply pca().
- We will use the bio3d pca() function which is designed for protein structure data.

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



^ Results of PCA on Adenylate kinase X-ray structures. Each dot represents one PDB structure.

5. Optional further visualization

Make a trajectory visualization of the motion captured by the first Principal Component

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

^Can open up the file and view in VMD, same as before

```
#![](pc.1.movie.png)
```

Rmd will not knit with the insertion of png image so I put it as a comment in a code chunk…

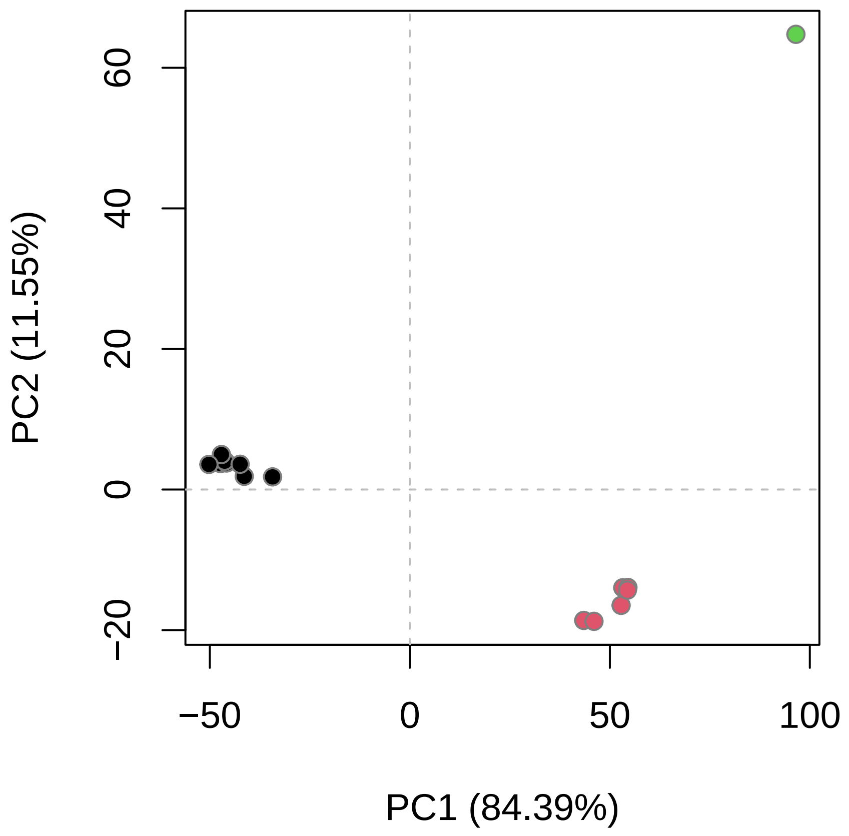
Function rmsd() will calculate all pairwise RMSD values of the structural ensemble. This facilitates clustering analysis based on the pairwise structural deviation:

```
# 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)
```



ˆProjection of Adenylate kinase X-ray structures. Each dot represents one PDB structure

- The plot shows a conformer plot – a low-dimensional representation of the conformational variability within the ensemble of PDB structures. The plot is obtained by projecting the individual structures onto two selected PCs (e.g. PC-1 and PC-2). These projections display the inter-conformer relationship in terms of the conformational differences described by the selected PCs.

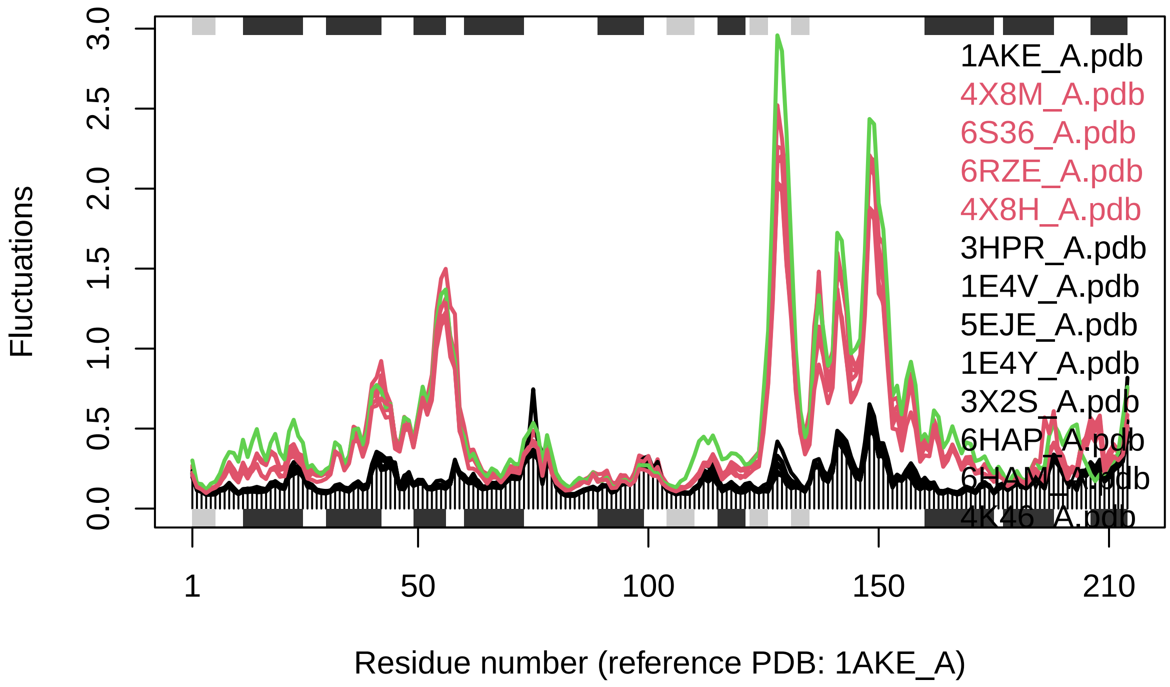
6. Normal mode analysis

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

```
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 and colored lines are different. They differ most at residues 125 - 150, likely because they have the most divergent structures at these locations.

Collectively these results indicate the existence of two major distinct conformational states for Adk. These differ by a collective low frequency displacement of two nucleotide-binding site regions that display distinct flexibilities upon nucleotide binding.