

Class 10: Structural Bioinformatics

Dylan Mullaney (A16869792)

Table of contents

1. Introduction to the RCSB Protein Data Bank (PDB)	1
2. Using Mol*	4
3. Intro to Bio3D in R	8
4. Predicting function dynamics	10
4. PCA of Adenylate Kinase (Adk)	12
Normal mode analysis	20

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

The main repository of biomolecular structure data is called the PDB found at: <https://www.rcsb.org/>

Let's see what this database contains. I went to PDB > Analyze > PDB Statistics > By Experimental method and molecular type

```
pdbstats <- read.csv ("Data Export Summary.csv")
pdbstats
```

	Molecular.Type	X.ray	EM	NMR	Multiple.methods	Neutron	Other
1	Protein (only)	169,563	16,774	12,578	208	81	32
2	Protein/Oligosaccharide	9,939	2,839	34	8	2	0
3	Protein/NA	8,801	5,062	286	7	0	0
4	Nucleic acid (only)	2,890	151	1,521	14	3	1
5	Other	170	10	33	0	0	0
6	Oligosaccharide (only)	11	0	6	1	0	4
	Total						
1		199,236					
2		12,822					

```

3 14,156
4 4,580
5 213
6 22

```

Looks like columns with numbers over 999 have become chr due to the commas. We cannot do math with these values - they must be a numeric type. I can fix this by replacing “,” for nothings “” with the `sub()` function:

```

#x<- pdbstats$X.ray
#sum (as.numeric(sub(",", "", x)))

```

Or I can use the **readr** package and the `read_csv()` function.

```

library (readr)
pdbstats <- read_csv ("Data Export Summary.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.

```

pdbstats

```

A tibble: 6 x 8

	<code>`Molecular Type`</code>	<code>`X-ray`</code>	EM	NMR	<code>`Multiple methods`</code>	Neutron	Other	Total
	<code><chr></code>	<code><dbl></code>	<code><dbl></code>	<code><dbl></code>	<code><dbl></code>	<code><dbl></code>	<code><dbl></code>	<code><dbl></code>
1	Protein (only)	169563	16774	12578	208	81	32	199236
2	Protein/Oligosacc~	9939	2839	34	8	2	0	12822
3	Protein/NA	8801	5062	286	7	0	0	14156
4	Nucleic acid (onl~	2890	151	1521	14	3	1	4580
5	Other	170	10	33	0	0	0	213
6	Oligosaccharide (~	11	0	6	1	0	4	22

I want to clean up the column names so they’re more consistently capitalized.

```
library (janitor)
```

Attaching package: 'janitor'

The following objects are masked from 'package:stats':

chisq.test, fisher.test

```
df <- clean_names (pdbstats)
```

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

83%, 11%

Total # of xray, em

```
sum(df$x_ray)
```

```
[1] 191374
```

```
sum(df$em)
```

```
[1] 24836
```

Total number of structures

```
sum(df$total)
```

```
[1] 231029
```

Percentage of X ray

```
(sum(df$x_ray) / sum(df$total)) *100
```

```
[1] 82.83549
```

Percentage of electron microscopy

```
(sum(df$em) / sum(df$total)) *100
```

```
[1] 10.75017
```

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

86% of structures are protein

```
totalp <- df [1,8]  
totalp
```

```
# A tibble: 1 x 1  
  total  
  <dbl>  
1 199236
```

```
protein.structures <- (totalp/sum(df$total))*100  
protein.structures
```

```
total  
1 86.23852
```

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?

Skipped

2. Using Mol*

The main Mol* homepage is at: <https://molstar.org/viewer/> We can input our own PDB files or just give it a PDB accession code (the 4 letter PDB code).

I took a screenshot on Mol* and downloaded it as a png. Let's use markdown code to import it: ! [] () -> caption in square, location/title in smooth, must have dragged image to folder



Figure 1: Molecular View of 1HSG

More images with more specificity

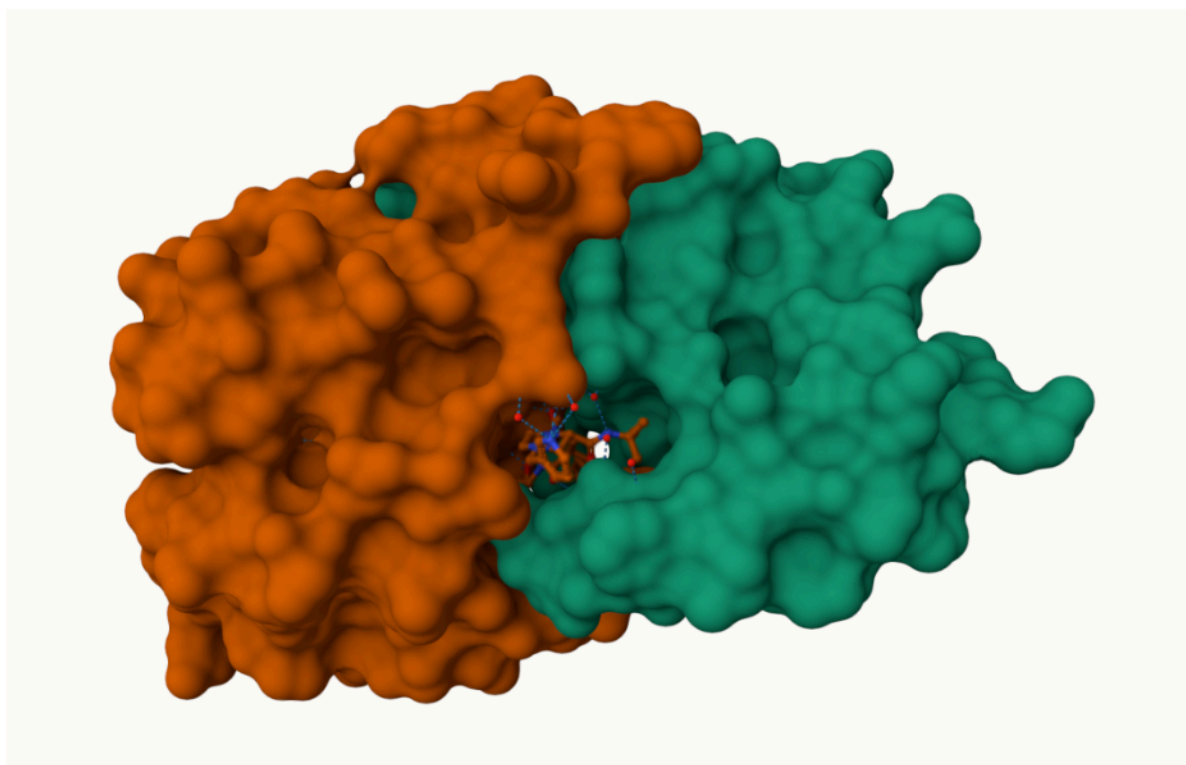


Figure 2: Molecular Binding Site

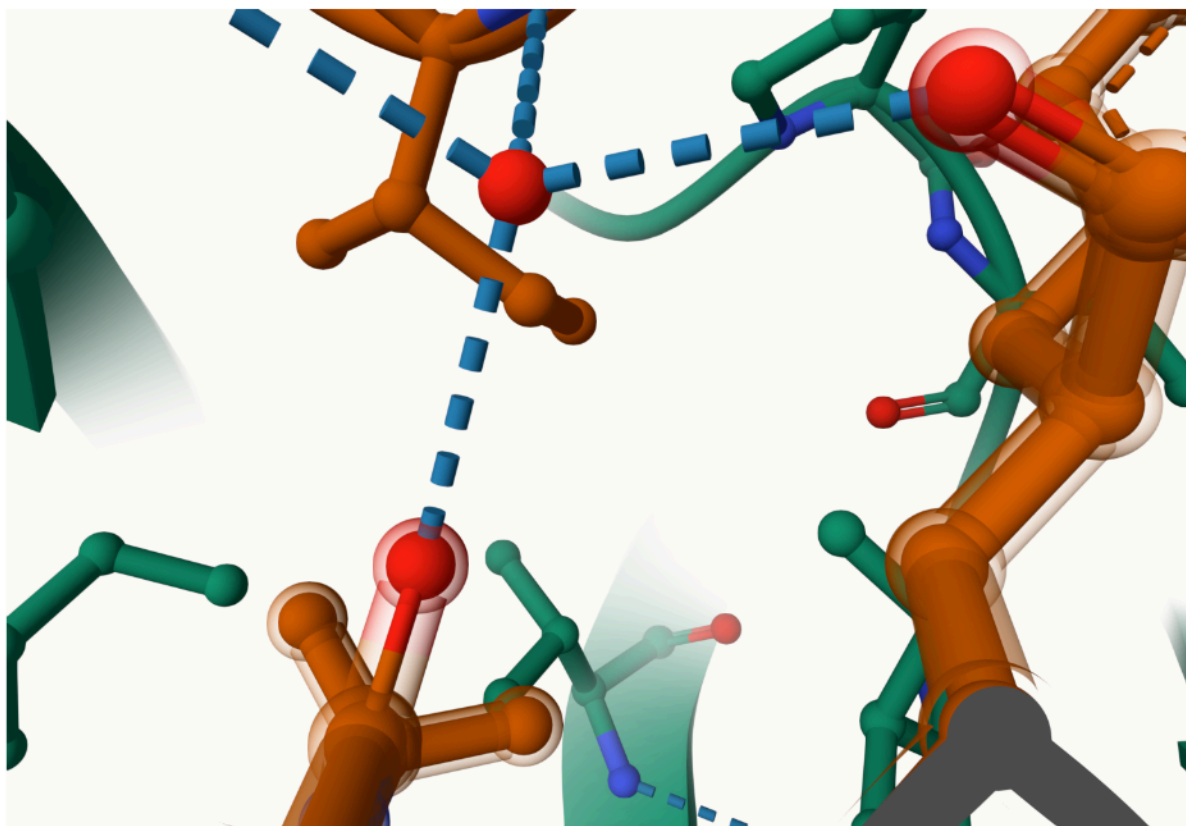


Figure 3: Conserved water in binding site

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

The hydrogen bonds are not shown. The presence of hydrogen is generally assumed and not explicitly added. This simplifies the image and makes it more digestible without losing vital information/

Q5: There is a critical “conserved” water molecule in the binding site. Can you identify this water molecule? What residue number does this water molecule have?

This water molecule is shown in the “Conserved water in binding site” image above. This water is residue number 308.

Q6: 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 and the critical water (we recommend “Ball & Stick” for these side-chains). Add this figure to your Quarto document.

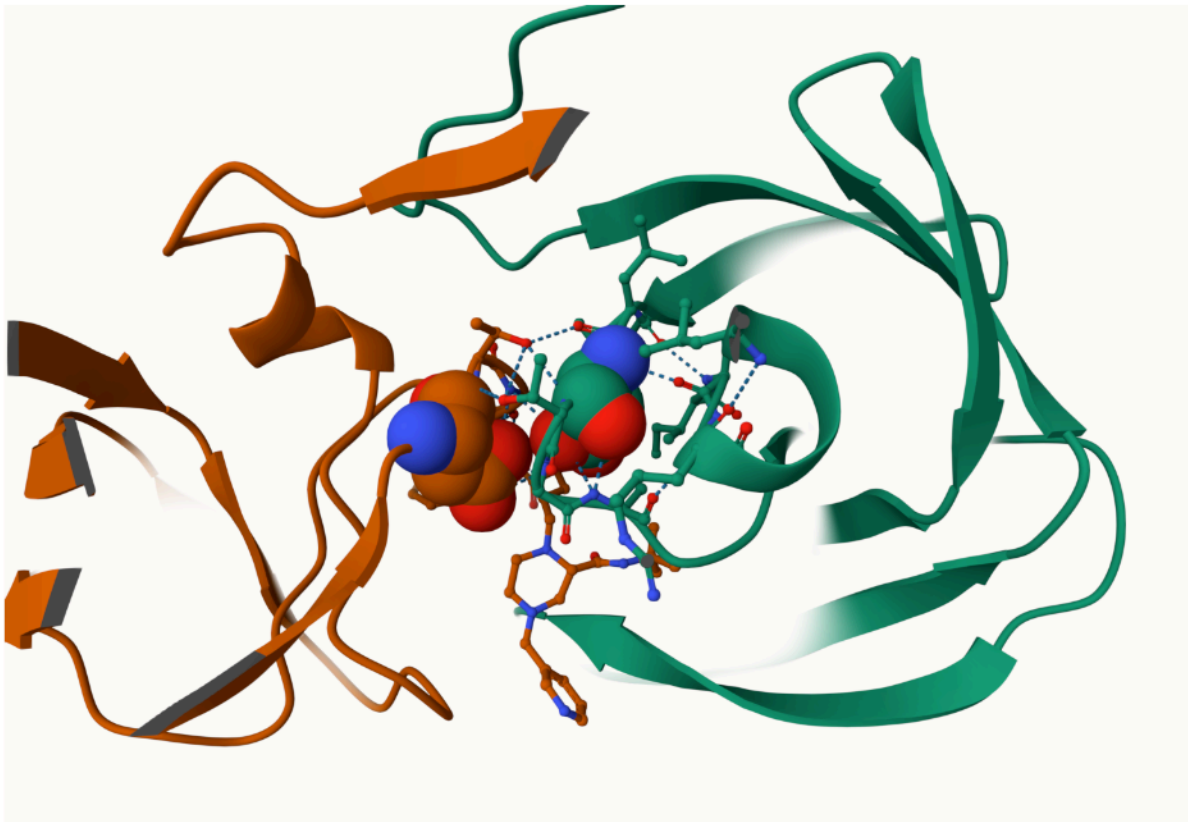


Figure 4: Two ASP shown in the binding sites

3. Intro to Bio3D in R

We can use the **bio3d** package for structural bioinformatics to read PDB data into R

```
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

```

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

198

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

```
[1] 198
```

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

Water, MK1 (ligand)

Q9: How many protein chains are in this structure?

Two chains (A, B)

Looking at pdb in more details

```
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>

Let's try a new function not yet in the bio3d package and install "r3dmol" and "shiny" in R console

```
source("https://tinyurl.com/viewpdb")
#view.pdb(pdb, backgroundColor = "lightyellow")
```

The purpose of the bio3d package is to be able to make predictions and analyses on the data imported.

4. Predicting function dynamics

We can use the `nma()` function in bio3d to predict the large-scale functional motions of biomolecules.

```
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:
```

```
MRIILLGAPGAGKGTQAQFIMEKYGIPQISTGDMRLRAAVKSGSELGKQAKDIMDAGKLV  
TDELVIALVKERIAQEDCRNGFLLDGFPRTIPQADAMKEAGINVDYVLEFDVPDELIVDKI  
VGRRVHAPSGRVYHVKFNPVKVEGKDDVTGEELTTRKDDQEETVRKRLVEYHQMTAPLIG  
YYSKEAEAGNTKYAKVDGTPVAEVRADLEKILG
```

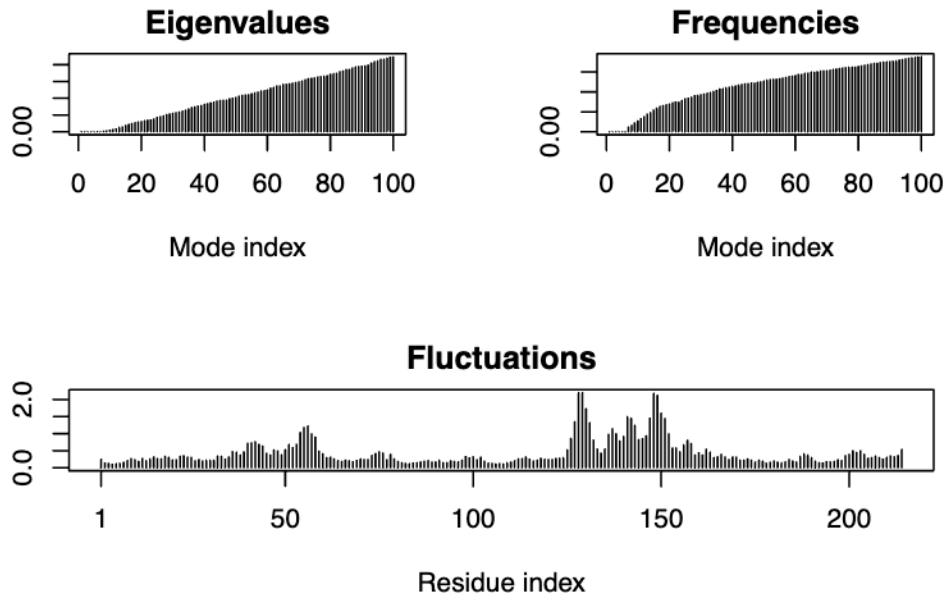
```
+ attr: atom, xyz, seqres, helix, sheet,  
      calpha, remark, call
```

```
m<- nma(adk)
```

```
Building Hessian... Done in 0.03 seconds.
```

```
Diagonalizing Hessian... Done in 0.581 seconds.
```

```
plot(m)
```



Make a forward trajectory of the predicted molecular motion

```
mktrj(m, file="adk_m7.pdb")
```

4. PCA of Adenylate Kinase (Adk)

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?:

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 -> get seq and blast pdb

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

```
Warning in get.seq("lake_A"): Removing existing file: seqs.fasta
```

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

```
aa
```

```

      1      .      .      .      .      .      60
pdb|1AKE|A  MRIILLGAPGAGKGTQAQFIMEKYGIPQISTGDMLRAAVKSGSELGKQAKDIMDAGKLV
      1      .      .      .      .      .      60

      61      .      .      .      .      .      120
pdb|1AKE|A  DELVIALVKERIAQEDCRNGFLLDGFPRTPQADAMKEAGINVDYVLEFDVPDELIVDRI
      61      .      .      .      .      .      120

     121      .      .      .      .      .      180
pdb|1AKE|A  VGRRVHAPSGRVYHVKNPPKVEGKDDVTGEELTRKDDQEETVRKRLVEYHQMTPALIG
     121      .      .      .      .      .      180

     181      .      .      .      214
pdb|1AKE|A  YYSKEAEAGNTKYAKVDGTPVAEVRADLEKILG
     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

Use this sequence as a query for BLAST

```
#b <- blast.pdb(aa)
```

```
#hits <- plot(b)
```

Top hits

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

```
hits <- NULL
```

```
hits$pdb.id <- c('1AKE_A', '6S36_A', '6RZE_A', '3HPR_A', '1E4V_A', '5EJE_A', '1E4Y_A', '3X2S_A', '6HPR_A')
```

Download related 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/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/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
```

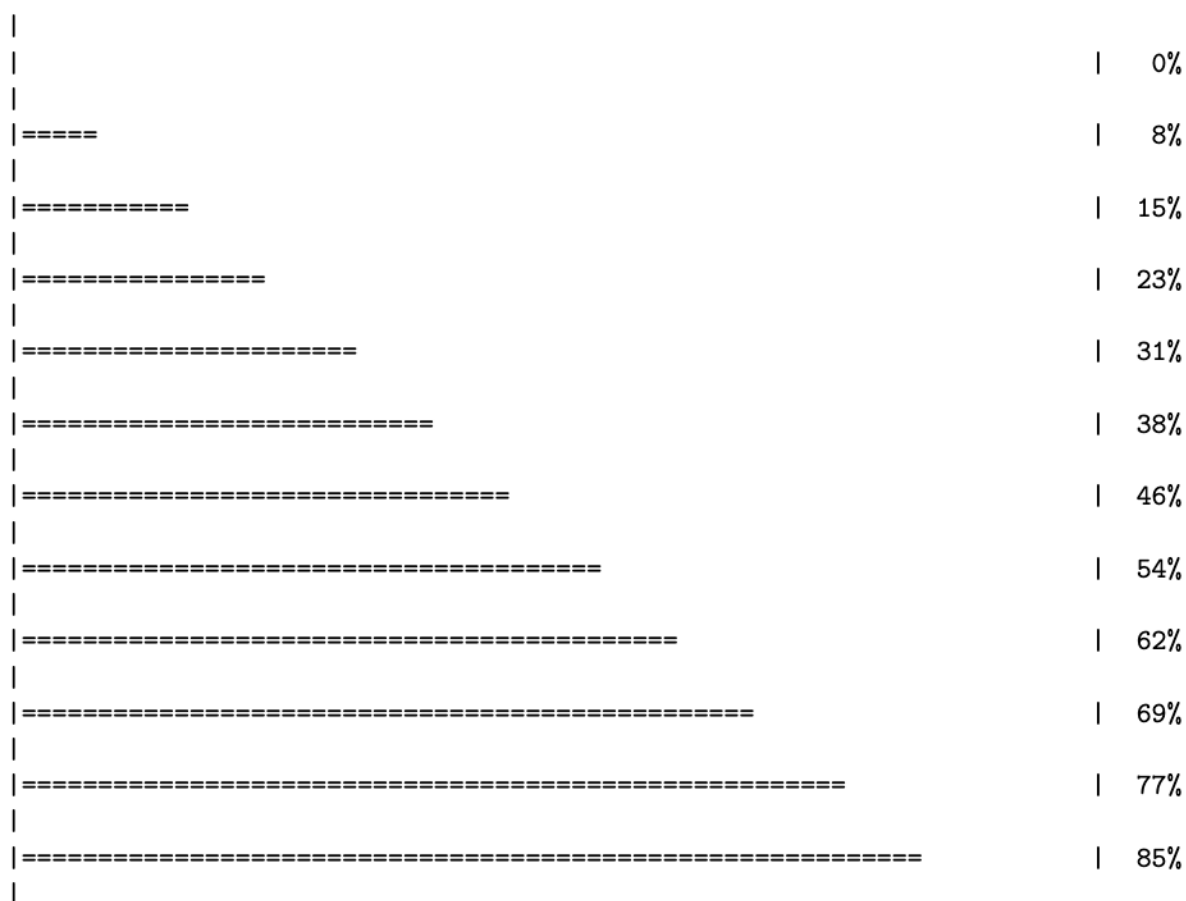
Warning in get.pdb(hits\$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE):
pdbs/6HAP.pdb.gz exists. Skipping download

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/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



```

|=====| 92%
|
|=====| 100%

```

Align and superpose structures

```
pdbbs <- pdbaln(files, fit = TRUE, exefile="msa")
```

Reading PDB files:

```

pdbbs/split_chain/1AKE_A.pdb
pdbbs/split_chain/6S36_A.pdb
pdbbs/split_chain/6RZE_A.pdb
pdbbs/split_chain/3HPR_A.pdb
pdbbs/split_chain/1E4V_A.pdb
pdbbs/split_chain/5EJE_A.pdb
pdbbs/split_chain/1E4Y_A.pdb
pdbbs/split_chain/3X2S_A.pdb
pdbbs/split_chain/6HAP_A.pdb
pdbbs/split_chain/6HAM_A.pdb
pdbbs/split_chain/4K46_A.pdb
pdbbs/split_chain/3GMT_A.pdb
pdbbs/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: pdbbs/split_chain/1AKE_A.pdb
  PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 2  name: pdbbs/split_chain/6S36_A.pdb
  PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 3  name: pdbbs/split_chain/6RZE_A.pdb
  PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 4  name: pdbbs/split_chain/3HPR_A.pdb
  PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 5  name: pdbbs/split_chain/1E4V_A.pdb

```



```

pdb/seq: 6   name: pdbs/split_chain/5EJE_A.pdb
  PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 7   name: pdbs/split_chain/1E4Y_A.pdb
pdb/seq: 8   name: pdbs/split_chain/3X2S_A.pdb
pdb/seq: 9   name: pdbs/split_chain/6HAP_A.pdb
pdb/seq: 10  name: pdbs/split_chain/6HAM_A.pdb
  PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 11  name: pdbs/split_chain/4K46_A.pdb
  PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 12  name: pdbs/split_chain/3GMT_A.pdb
pdb/seq: 13  name: pdbs/split_chain/4PZL_A.pdb

```

```

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

```

```

# Draw schematic alignment

```

```

#plot(pdb, labels=ids)

```

```

anno <- pdb.annotate(ids)
unique(anno$source)

```

```

[1] "Escherichia coli"
[2] "Escherichia coli K-12"
[3] "Escherichia coli 0139:H28 str. E24377A"
[4] "Escherichia coli str. K-12 substr. MDS42"
[5] "Photobacterium profundum"
[6] "Burkholderia pseudomallei 1710b"
[7] "Francisella tularensis subsp. tularensis SCHU S4"

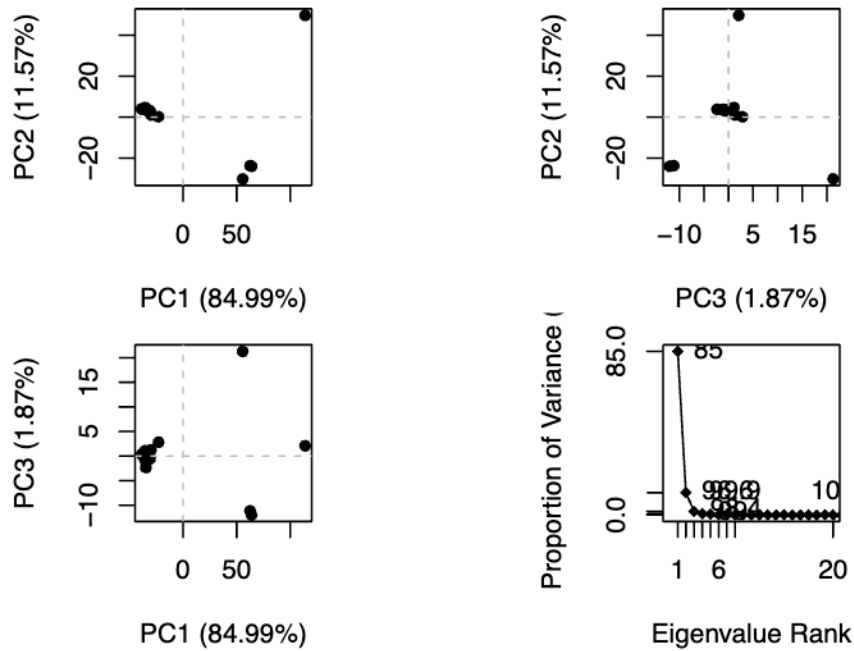
```

Principal Component Analysis

```

pc.xray <- pca (pdb)
plot (pc.xray)

```



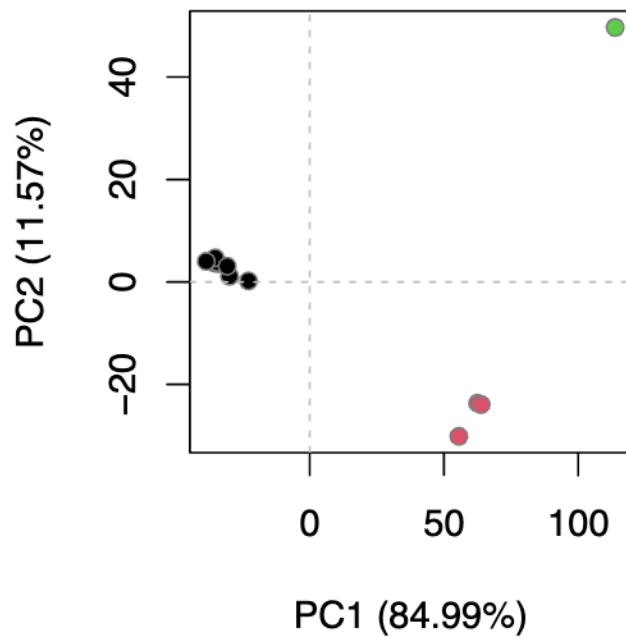
The function `rmsd()` will calculate pairwise RMSD values -> facilitating clustering analysis based on pairwise structural deviation

```
rd <- rmsd(pdbbs)
```

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

```
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)
```



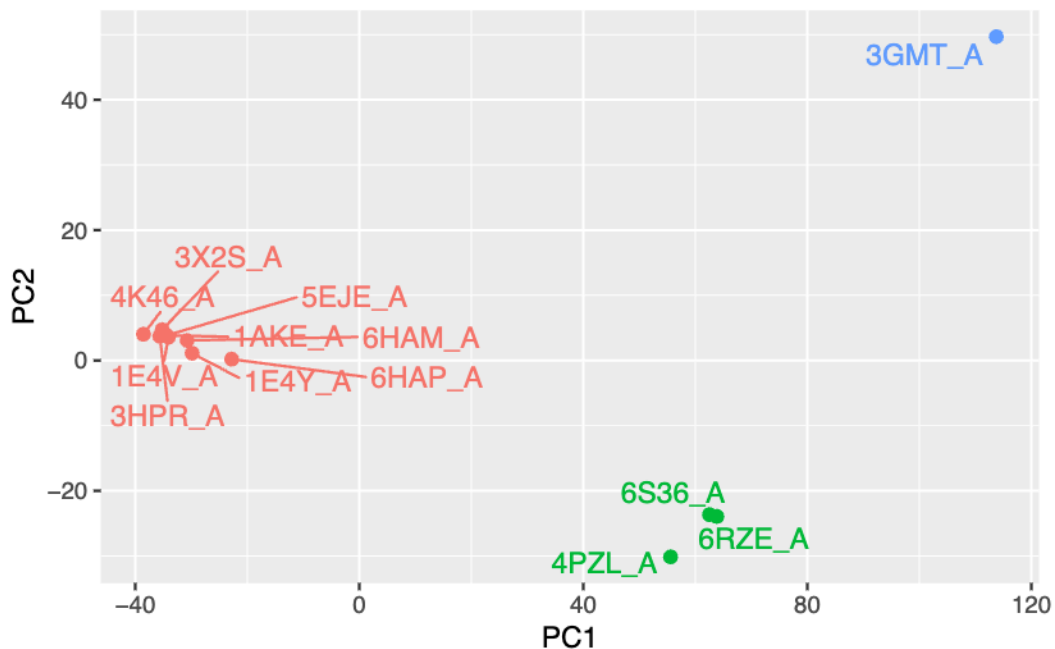
This is a conformer plot → low density representation of the conformation variability within the PDB structures

Plot PCA results with ggplot

```
library(ggplot2)
library(ggrepel)

df <- data.frame(PC1=pc.xray$z[,1],
                  PC2=pc.xray$z[,2],
                  col=as.factor(grps.rd),
                  ids=ids)

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
```



Normal mode analysis

```
modes <- nma (pddb)
```

Details of Scheduled Calculation:

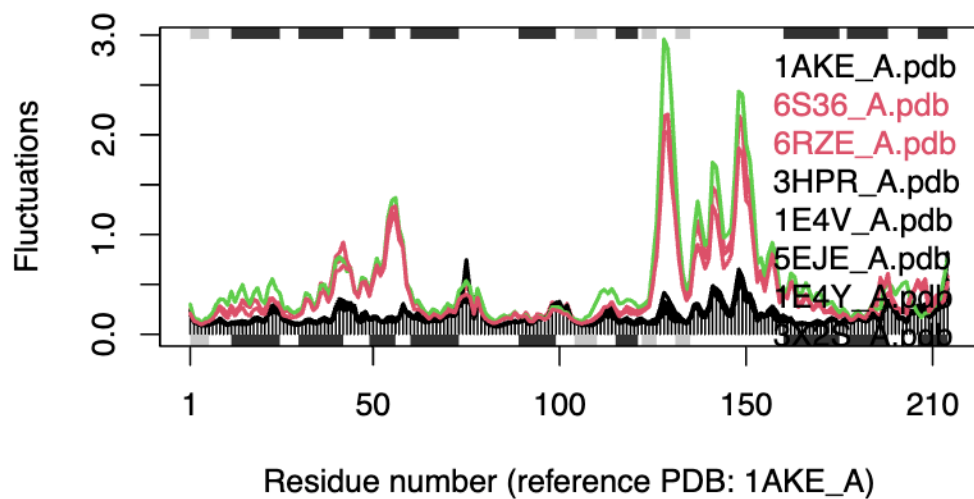
```
... 13 input structures
... storing 606 eigenvectors for each structure
... dimension of x$U.subspace: ( 612x606x13 )
... coordinate superposition prior to NM calculation
... aligned eigenvectors (gap containing positions removed)
... estimated memory usage of final 'eNMA' object: 36.9 Mb
```

			0%
=====			8%
=====			15%



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

Extracting SSE from pdbc\$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?

Both the red and green lines follow a very similar pattern, with the green line trending above the red one. The black line falls significantly under the green and red lines but in a similar shape.