Lab_09

RUNQI ZHANG

Table of contents

The RCSB Protein Data Bank (PDB)

Download a CSV file from the PDB site **(accessible from "Analyze" > "PDB Statistics" > "by Experimental Method and Molecular Type"**. Move this CSV file into your RStudio project and use it to answer the following questions:

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

```
           Molecular.Type   X.ray    NMR    EM Multiple.methods Neutron Other
1          Protein (only) 150,342 12,053 8,534              188      72    32
2 Protein/Oligosaccharide   8,866     32 1,540                6       0     0
3              Protein/NA   7,911    278 2,681                6       0     0
4     Nucleic acid (only)   2,510  1,425    74               13       2     1
5                   Other     154     31     6                0       0     0
6  Oligosaccharide (only)      11      6     0                1       0     4
    Total
1 171,221
2  10,444
3  10,876
4   4,025
5     191
6      22
```

```
n_xray <- sum(strtoi(gsub(",", "", PDB_stat$X.ray)))
n_xray
```

```
[1] 169794
```

```
n_total <- sum(strtoi(gsub(",", "", PDB_stat$Total)))
n_total
```

```
[1] 196779
```

```
n_xray/n_total
```

```
[1] 0.8628665
```

```
n_protein <- PDB_stat$Total[1]
n_protein <- strtoi(gsub(",", "",n_protein))
n_protein/n_total
```

```
[1] 0.8701183
```

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

A: > n_xray/n_total [1] 0.8628665

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

A: > n_protein/n_total [1] 0.8701183

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?

A: 272 ## access denied due to slow network, come back later

https://molstar.org/viewer/

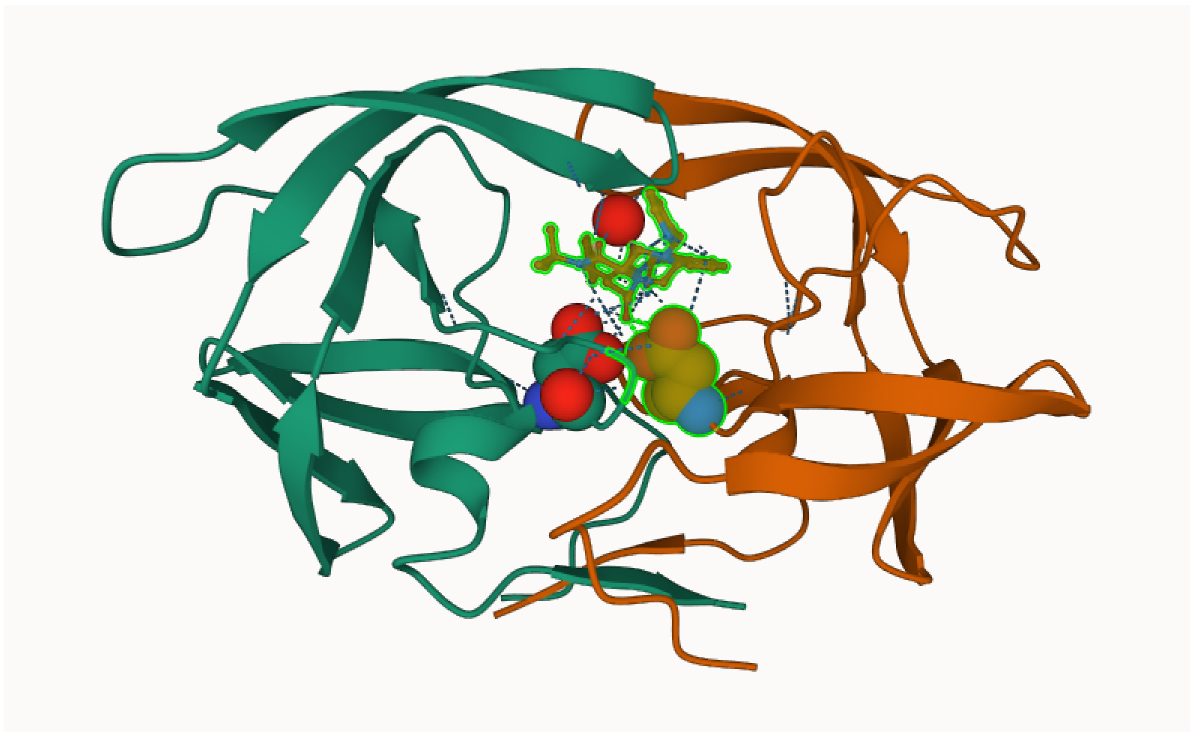
Q4: Water molecules normally have 3 atoms. Why do we see just one atom per water molecule in this structure?

A: water molecules contain three atoms: one oxygen and two hydrogens. The hydrogen atom is the first on the periodic table and has the smallest size. With the resolution applied for the purpose of crystalography, hydrogen molecule could not be detected.

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

A: The water molecule is found between the ligand and the isoleucine on residue 50.

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 (we recommend “Ball & Stick” for these side-chains). Add this figure to your Quarto document.



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

Q7: [Optional] 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 can you identify secondary structure elements that are likely to only form in the dimer rather than the monomer?

```r
library(bio3d)
pdb <- read.pdb("1hsg")
```

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

```r
pdf
```

```
function (file = if (onefile) "Rplots.pdf" else "Rplot%03d.pdf",
    width, height, onefile, family, title, fonts, version, paper,
    encoding, bg, fg, pointsize, pagecentre, colormodel, useDingbats,
    useKerning, fillOddEven, compress)
{
    initPSandPDFfonts()
    new <- list()
    if (!missing(width))
        new$width <- width
    if (!missing(height))
        new$height <- height
    if (!missing(onefile))
        new$onefile <- onefile
    if (!missing(title))
        new$title <- title
    if (!missing(fonts))
        new$fonts <- fonts
    if (!missing(version))
        new$version <- version
    if (!missing(paper))
        new$paper <- paper
    if (!missing(encoding))
        new$encoding <- encoding
    if (!missing(bg))
        new$bg <- bg
    if (!missing(fg))
        new$fg <- fg
    if (!missing(pointsize))
        new$pointsize <- pointsize
    if (!missing(pagecentre))
        new$pagecentre <- pagecentre
    if (!missing(colormodel))
        new$colormodel <- colormodel
```

```
if (!missing(useDingbats))
    new$useDingbats <- useDingbats
if (!missing(useKerning))
    new$useKerning <- useKerning
if (!missing(fillOddEven))
    new$fillOddEven <- fillOddEven
if (!missing(compress))
    new$compress <- compress
old <- check.options(new, name.opt = ".PDF.Options", envir = .PSenv)
if (!missing(family) && (inherits(family, "Type1Font") ||
    inherits(family, "CIDFont"))) {
    enc <- family$encoding
    if (inherits(family, "Type1Font") && !is.null(enc) &&
        enc != "default" && (is.null(old$encoding) || old$encoding ==
        "default"))
        old$encoding <- enc
    family <- family$metrics
}
if (is.null(old$encoding) || old$encoding == "default")
    old$encoding <- guessEncoding()
if (!missing(family)) {
    if (length(family) == 4L) {
        family <- c(family, "Symbol.afm")
    }
    else if (length(family) == 5L) {
    }
    else if (length(family) == 1L) {
        pf <- pdfFonts(family)[[1L]]
        if (is.null(pf))
            stop(gettextf("unknown family '%s'", family),
              domain = NA)
        matchFont(pf, old$encoding)
    }
    else stop("invalid 'family' argument")
    old$family <- family
}
version <- old$version
versions <- c("1.1", "1.2", "1.3", "1.4", "1.5", "1.6", "1.7",
    "2.0")
if (version %in% versions)
    version <- as.integer(strsplit(version, "[.]")[[1L]])
else stop("invalid PDF version")
onefile <- old$onefile
```

```
    if (!checkIntFormat(file))
        stop(gettextf("invalid 'file' argument '%s'", file),
            domain = NA)
    .External(C_PDF, file, old$paper, old$family, old$encoding,
        old$bg, old$fg, old$width, old$height, old$pointsize,
        onefile, old$pagecentre, old$title, old$fonts, version[1L],
        version[2L], old$colormodel, old$useDingbats, old$useKerning,
        old$fillOddEven, old$compress)
    invisible()
}
<bytecode: 0x000001b75bc9c630>
<environment: namespace:grDevices>
```

The ATOM records of a PDF file are stored in ‘pdf$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>
```

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

A: 198

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

A: MK1

Q9: How many protein chains are in this structure?

A: 2

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

A: MSA

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

A: bio3d-view

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

A: TRUE

Use these ADK structures for analysis

```
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
```

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

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

# 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/
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/
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/
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%
  |
  |=====                                                                 |   8%
  |
  |===========                                                           |  15%
  |
  |================                                                      |  23%
  |
  |======================                                                |  31%
  |
  |===========================                                           |  38%
  |
  |================================                                      |  46%
  |
  |======================================                                |  54%
  |
  |===========================================                           |  62%
  |
  |================================================                      |  69%
  |
  |======================================================                |  77%
  |
  |===========================================================           |  85%
  |
  |=================================================================     |  92%
  |
  |======================================================================| 100%
```

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

A: 214

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

```
Reading PDB files:
pdbs/split_chain/1AKE_A.pdb
pdbs/split_chain/6S36_A.pdb
pdbs/split_chain/6RZE_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/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/6S36_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 3   name: pdbs/split_chain/6RZE_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 4   name: pdbs/split_chain/3HPR_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 5   name: pdbs/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
```

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

# Draw schematic alignment
#plot(pdbs, labels=ids)
#figure margin too large, cause issues during rendering
```

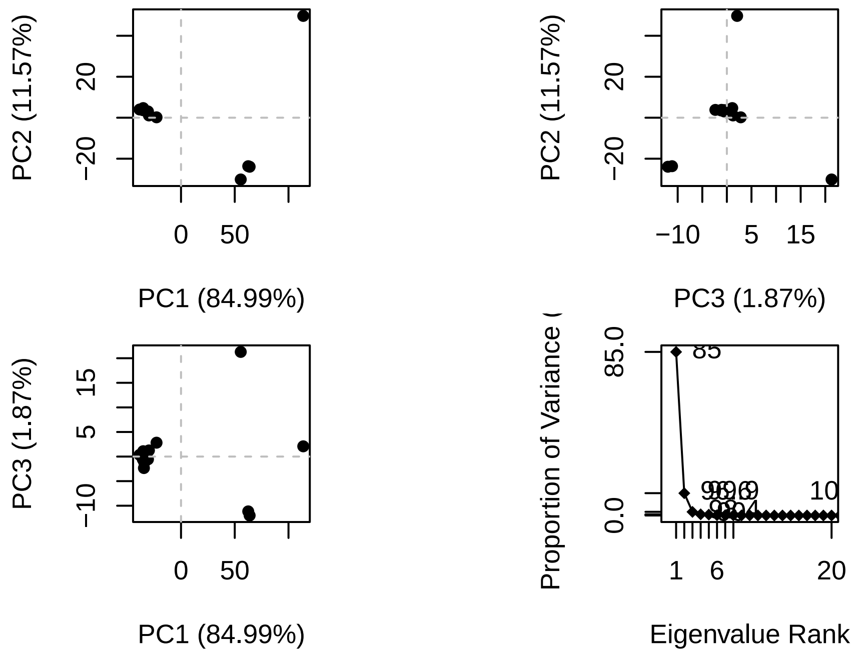
```r
anno <- pdb.annotate(ids)
unique(anno$source)
```

```
[1] "Escherichia coli"
```

```
[2] "Escherichia coli K-12"
[3] "Escherichia coli O139: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"
```

1. get.seq()
2. blast.pbd()
3. get.pdb(hits$pdb.id, path="pdbs", split=TRUE, gzip=TRUE)
4. pdbaln(files, fit = TRUE, exefile="msa")

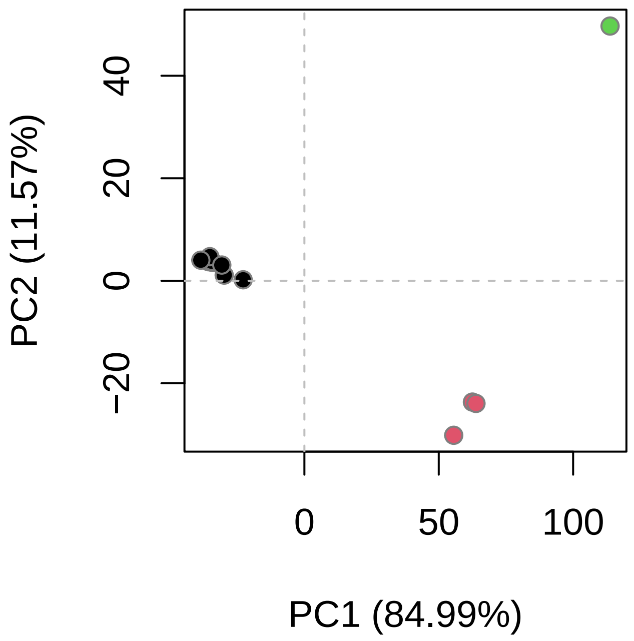
```
# Perform PCA
pc.xray <- pca(pdbs)
plot(pc.xray)
```



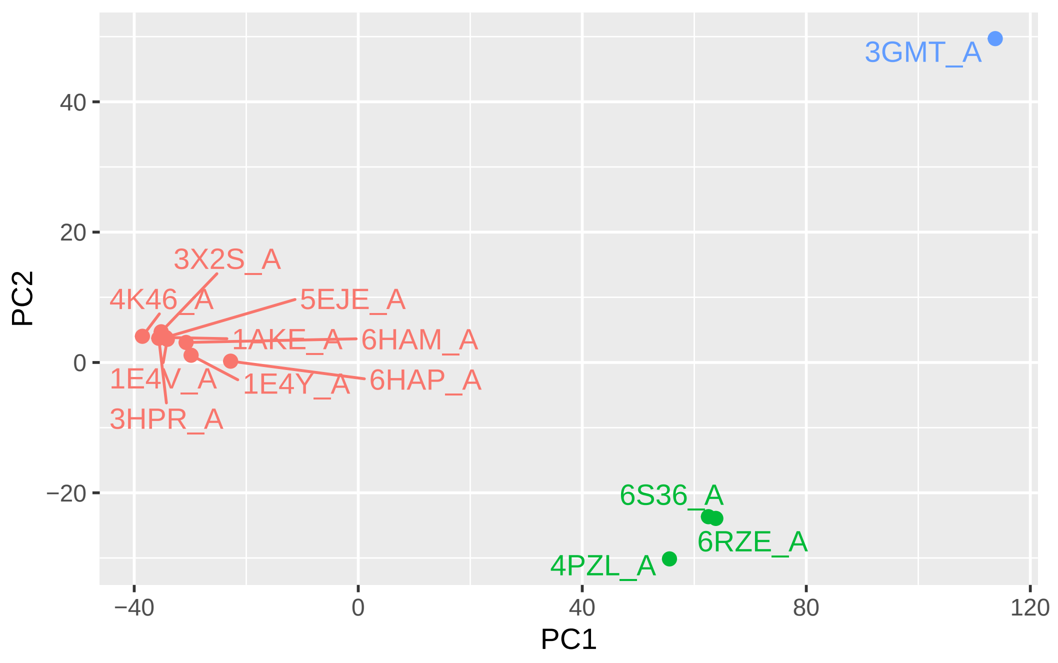
```
# Calculate RMSD
rd <- rmsd(pdbs)
```

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

```r
# 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)
```



```r
#Plotting results with ggplot2
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
```

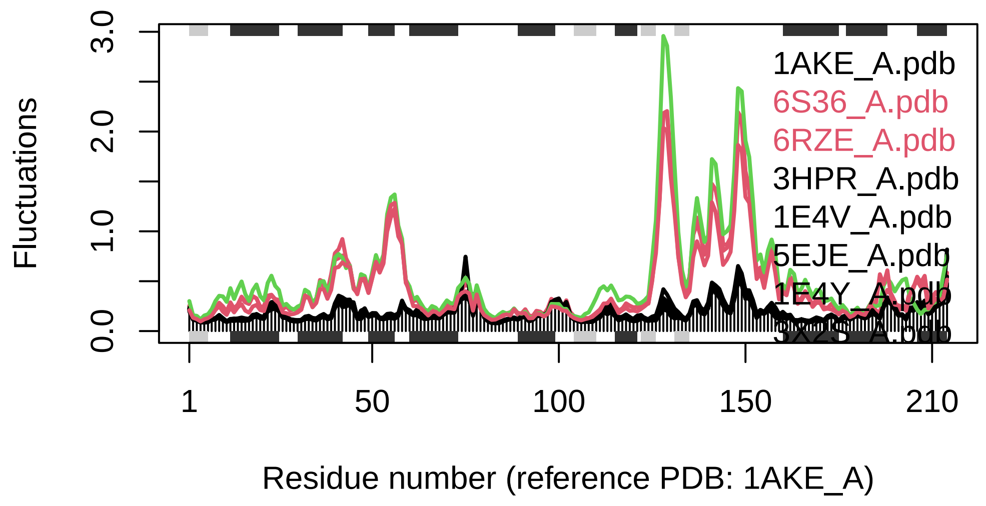


```
# NMA of all structures
modes <- nma(pdbs)
```

```
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%
  |
  |================                                                      |  23%
  |
  |======================                                                |  31%
  |
  |===========================                                           |  38%
  |
  |================================                                      |  46%
  |
  |======================================                                |  54%
  |
  |===========================================                           |  62%
  |
  |================================================                      |  69%
  |
  |======================================================                |  77%
  |
  |===========================================================           |  85%
  |
  |=================================================================     |  92%
  |
  |======================================================================| 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?

A: The black and colored lines are different and can be distinguished into two major categories. The difference is reflected in the proteins' flexibility and essentially suggests that there are more than one conformational states for the given protein.

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