Class10 structural bioinformatics (pt1)

Hanhee Jo (PID# A59017994)

Table of contents

The PDB satabase

The main repository for biomolecular data is called the PDB (Protein Data Bank) and can be found at: https://www.rcsb.org/

Let's see what it contains in terms of type of molecule and method of structure determination (Analyze > PDB stats > By Mol Type and Method)

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

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

```
nocomma <- sub(",", "", pdbstats$X.ray)
sum( as.numeric(nocomma) )
```

```
[1] 191374
```

Let's try the **readr** package and it's newer `read_csv()` function.

```
library(readr)

pbdstats <- 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.
```

```
pbdstats
```

```
# A tibble: 6 x 8
  `Molecular Type`   `X-ray`    EM   NMR `Multiple methods` Neutron Other  Total
  <chr>                <dbl> <dbl> <dbl>              <dbl>   <dbl> <dbl>  <dbl>
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
```

The resulting columnnames are "untidy" with spaces and a mix of upper and lower case letters that will make working with the columns a pain. We can use the **janitor** package qand it's `clean_names()` function to fix this for us.

```
colnames(pbdstats)
```

```
[1] "Molecular Type"   "X-ray"            "EM"               "NMR"
[5] "Multiple methods" "Neutron"          "Other"            "Total"
```

```
library(janitor)
```

```
Warning: package 'janitor' was built under R version 4.3.3
```

```
Attaching package: 'janitor'

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

    chisq.test, fisher.test
```

```
df <- clean_names(pbdstats)
```

Percent of structures in PDB solved by X-ray?

```
n.xray <- sum(df$x_ray)
n.total <- sum(df$total)
n.xray
```

```
[1] 191374
```

```
n.total
```

```
[1] 231029
```

In UniProt there are 253,206,171 protein sequences and there are only 231,029 known structures in the PDB. This is a tiny fraction!

```
231029/253206171 *100
```

```
[1] 0.09124146
```

Next day we will see how bioinformatics methods can help predict structure from sequence with accuracy approaching X-ray methods.

```
round(n.xray/n.total * 100, digits=2)
```

```
[1] 82.84
```

Percent of EM structures?

```
n.em <- sum(df$em)
n.total <- sum(df$total)
n.em
```

```
[1] 24836
```

```
n.total
```

```
[1] 231029
```

```
round(n.em/n.total * 100, digits=2)
```

```
[1] 10.75
```

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

```
round(df$total[1]/n.total * 100, digits=2)
```

```
[1] 86.24
```

2. Molecular visualization with Mol*

Mol-star is a new online structure viewer that is taking over the world of biomolecular visualization. Let's see how to use it from https://molstar.org/

My first image from Mol* of HIV-Pr

I want an image that shows the binding cleft for the MK1 inhibitor, an image of the most valuable water in human history, and an image showing the catalytic ASP amino-acids.

3. Using the Bio3D package

This package has tones of tools and utilities for structural bioinformatics.

```
library(bio3d)
```

```
Warning: package 'bio3d' was built under R version 4.3.3
```

Figure 1: Fig1. A first view of the HIV-Pr dimer

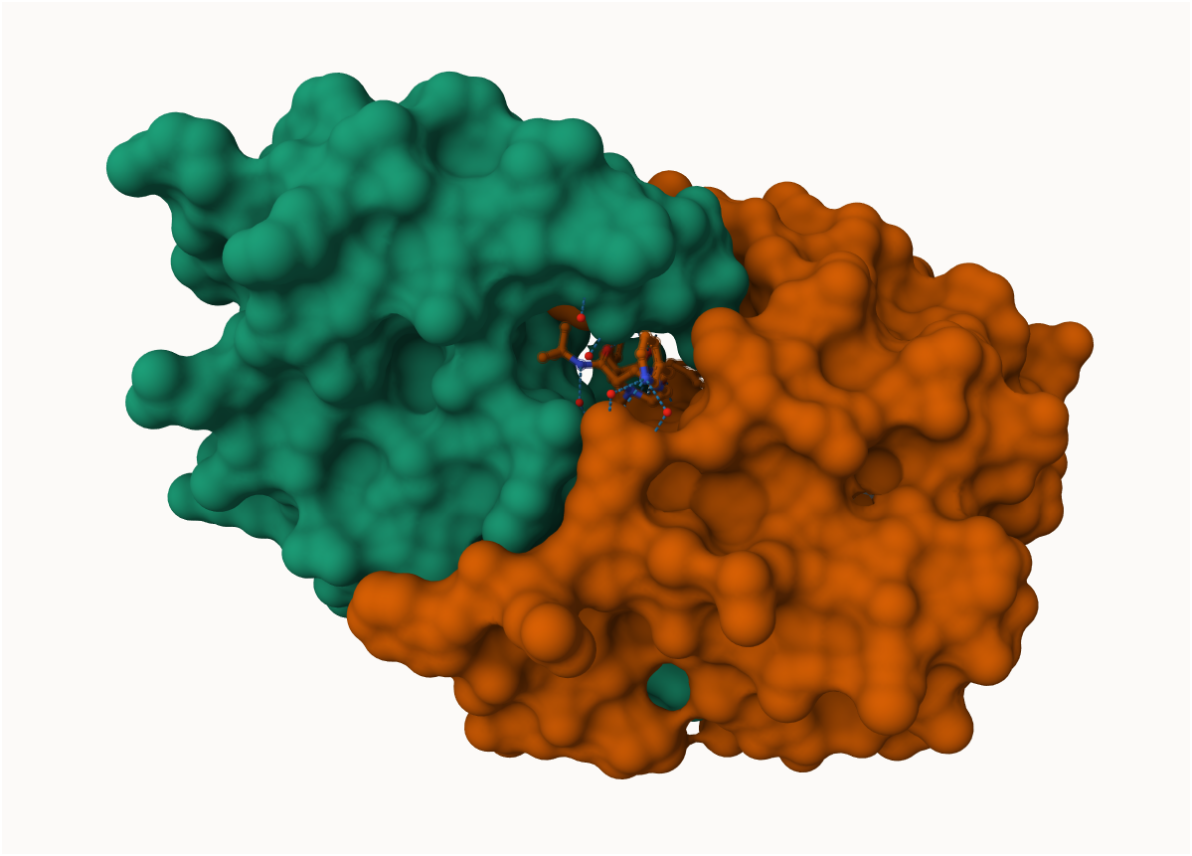


Figure 2: Binding cleft

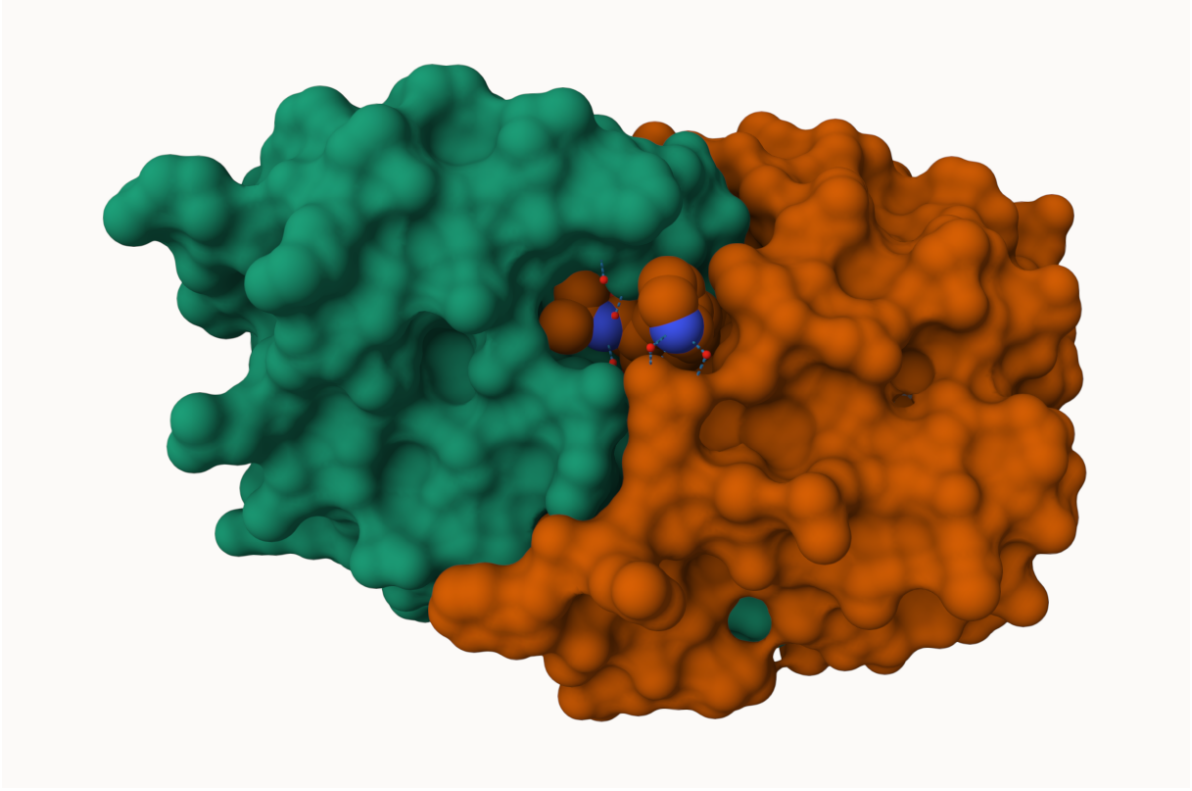


Figure 3: Ligand fill

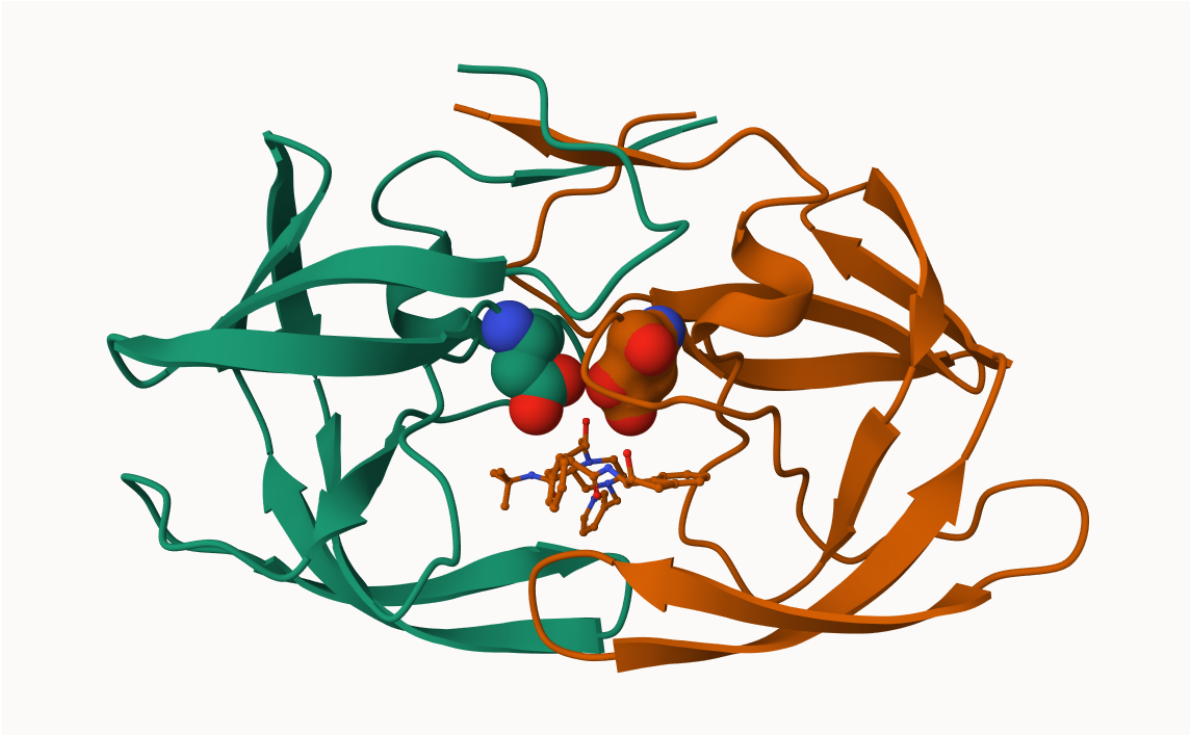


Figure 4: Overview with ASP25 highlighted

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

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

```
hiv
```

```
 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
```

```
head(hiv$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>
```

```
length( pdbseq(hiv) )
```

```
[1] 198
```

Predict functional motions

Let's read a new structure "6s36"

```
pdb <- read.pdb("6s36")
```

```
  Note: Accessing on-line PDB file
   PDB has ALT records, taking A only, rm.alt=TRUE
```

```
pdb
```

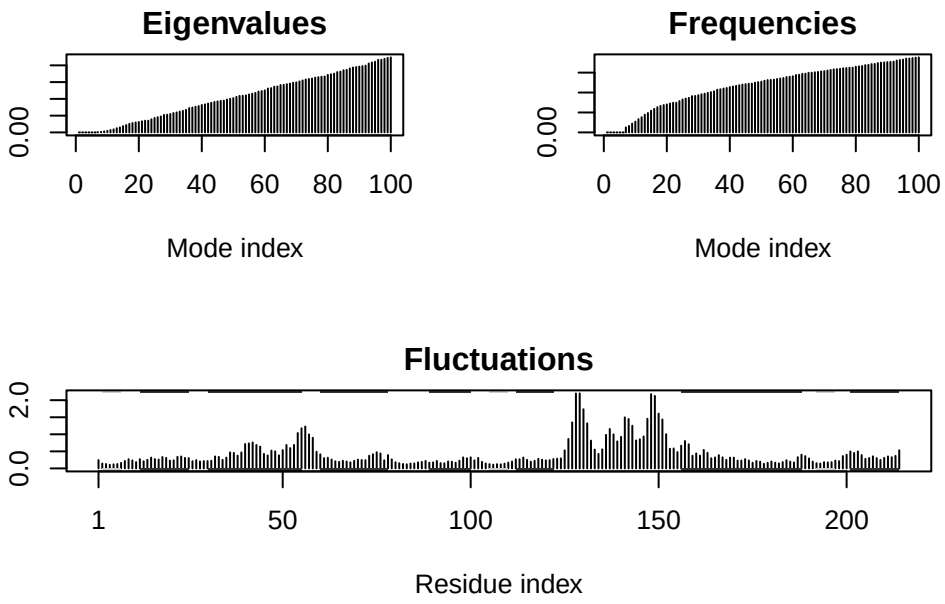
```
 Call:  read.pdb(file = "6s36")

   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:
      MRIILLGAPGAGKGTQAQFIMEKYGIPQISTGDMLRAAVKSGSELGKQAKDIMDAGKLVT
      DELVIALVKERIAQEDCRNGFLLDGFPRTIPQADAMKEAGINVDYVLEFDVPDELIVDKI
      VGRRVHAPSGRVYHVKFNPPKVEGKDDVTGEELTTRKDDQEETVRKRLVEYHQMTAPLIG
      YYSKEAEAGNTKYAKVDGTKPVAEVRADLEKILG

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

We can run a NMA calculation on this structure:

```
m <- nma(pdb)
```

```
Building Hessian...        Done in 0.013 seconds.
Diagonalizing Hessian...   Done in 0.286 seconds.
```

```
plot(m, sse = pdb)
```



We can write out a wee trajectory of the predicted dynamics using the `mktrj()`

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

Comparative analysis

```
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
```

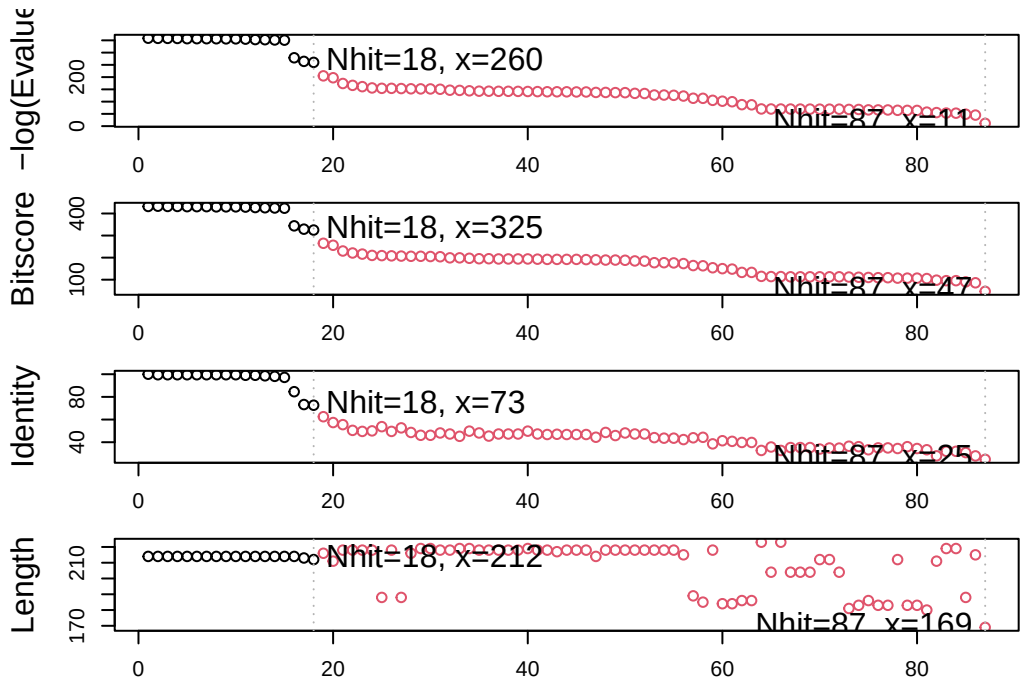
Search the PDB database for related sequences

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

```
 Searching ... please wait (updates every 5 seconds) RID = UD7RRCZ6013
 .....................
 Reporting 87 hits
```

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

```
* Possible cutoff values:    260 11
          Yielding Nhits:    18 87

* Chosen cutoff value of:    260
          Yielding Nhits:    18
```




```
head(blast$raw)
```

```
       queryid subjectids identity alignmentlength mismatches gapopens q.start
1 Query_5001169     1AKE_A  100.000             214          0        0       1
2 Query_5001169     8BQF_A   99.533             214          1        0       1
3 Query_5001169     4X8M_A   99.533             214          1        0       1
4 Query_5001169     6S36_A   99.533             214          1        0       1
5 Query_5001169     8Q2B_A   99.533             214          1        0       1
6 Query_5001169     8RJ9_A   99.533             214          1        0       1
  q.end s.start s.end    evalue bitscore positives
1   214       1   214 1.61e-156      432    100.00
2   214      21   234 2.64e-156      433    100.00
```

```
3   214       1   214 2.89e-156      432    100.00
4   214       1   214 4.24e-156      432    100.00
5   214       1   214 1.13e-155      431     99.53
6   214       1   214 1.13e-155      431     99.53
```

```
hits$pdb.id
```

```
 [1] "1AKE_A" "8BQF_A" "4X8M_A" "6S36_A" "8Q2B_A" "8RJ9_A" "6RZE_A" "4X8H_A"
 [9] "3HPR_A" "1E4V_A" "5EJE_A" "1E4Y_A" "3X2S_A" "6HAP_A" "6HAM_A" "8PVW_A"
[17] "4K46_A" "4NP6_A"
```

Download all these structures to our project dir

```
#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/8BQF.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/8Q2B.pdb.gz exists. Skipping download
```

```
Warning in get.pdb(hits$pdb.id, path = "pdbs", split = TRUE, gzip = TRUE):
pdbs/8RJ9.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/8PVW.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

  |
  |                                                                      |   0%
  |
  |====                                                                  |   6%
  |
  |========                                                              |  11%
  |
  |============                                                          |  17%
  |
```

```
  |================                                                      |  22%
  |
  |===================                                                   |  28%
  |
  |=======================                                               |  33%
  |
  |===========================                                           |  39%
  |
  |===============================                                       |  44%
  |
  |===================================                                   |  50%
  |
  |=======================================                               |  56%
  |
  |===========================================                           |  61%
  |
  |===============================================                       |  67%
  |
  |===================================================                   |  72%
  |
  |=======================================================               |  78%
  |
  |===========================================================           |  83%
  |
  |===============================================================       |  89%
  |
  |===================================================================   |  94%
  |
  |======================================================================| 100%
```

#Align and supperpose

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

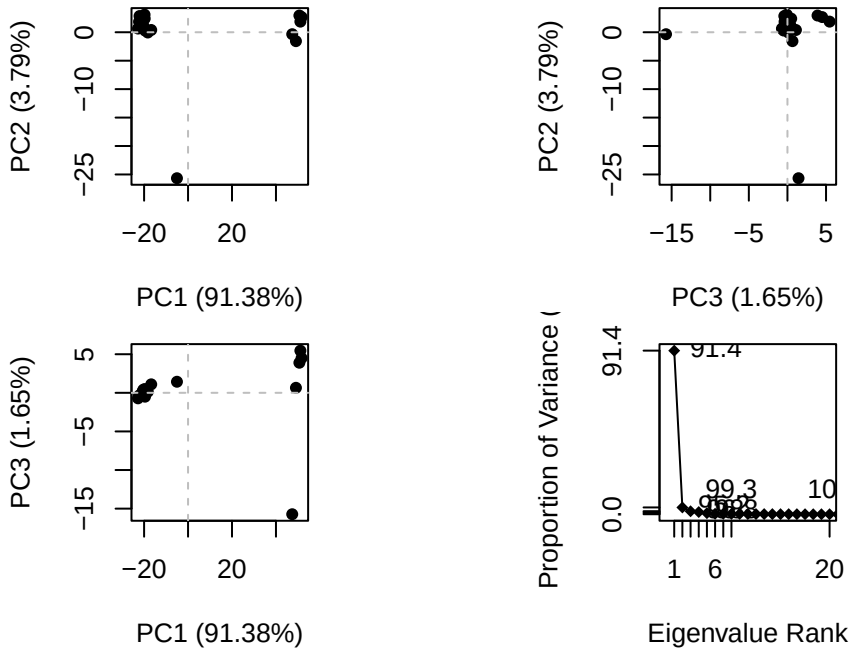
```
Reading PDB files:
pdbs/split_chain/1AKE_A.pdb
pdbs/split_chain/8BQF_A.pdb
pdbs/split_chain/4X8M_A.pdb
pdbs/split_chain/6S36_A.pdb
pdbs/split_chain/8Q2B_A.pdb
pdbs/split_chain/8RJ9_A.pdb
```

```
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/8PVW_A.pdb
pdbs/split_chain/4K46_A.pdb
pdbs/split_chain/4NP6_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
..   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/8BQF_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 3   name: pdbs/split_chain/4X8M_A.pdb
pdb/seq: 4   name: pdbs/split_chain/6S36_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 5   name: pdbs/split_chain/8Q2B_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 6   name: pdbs/split_chain/8RJ9_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 7   name: pdbs/split_chain/6RZE_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 8   name: pdbs/split_chain/4X8H_A.pdb
pdb/seq: 9   name: pdbs/split_chain/3HPR_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
```

```
pdb/seq: 10   name: pdbs/split_chain/1E4V_A.pdb
pdb/seq: 11   name: pdbs/split_chain/5EJE_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 12   name: pdbs/split_chain/1E4Y_A.pdb
pdb/seq: 13   name: pdbs/split_chain/3X2S_A.pdb
pdb/seq: 14   name: pdbs/split_chain/6HAP_A.pdb
pdb/seq: 15   name: pdbs/split_chain/6HAM_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 16   name: pdbs/split_chain/8PVW_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 17   name: pdbs/split_chain/4K46_A.pdb
   PDB has ALT records, taking A only, rm.alt=TRUE
pdb/seq: 18   name: pdbs/split_chain/4NP6_A.pdb
```

PCA

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



```r
plot(pc.xray, pc.axes = c(1,2))
```