Class 9: Structural Bioinformatics 1

Samson A16867000

The main database for structual data is called the PDB (Protein Data Bank). Let's see what it contains:

Data from: https://www.rcsb.org/stats

```
pdbdb <- read.csv("pdb_stats.csv")
pdbdb
```

```
           Molecular.Type   X.ray     EM    NMR Multiple.methods Neutron Other
1          Protein (only) 167,192 15,572 12,529              208      77    32
2 Protein/Oligosaccharide   9,639  2,635     34                8       2     0
3              Protein/NA   8,730  4,697    286                7       0     0
4     Nucleic acid (only)   2,869    137  1,507               14       3     1
5                   Other     170     10     33                0       0     0
6  Oligosaccharide (only)      11      0      6                1       0     4
    Total
1 195,610
2  12,318
3  13,720
4   4,531
5     213
6      22
```

```
pdbdb$Total
```

```
[1] "195,610" "12,318"  "13,720"  "4,531"   "213"     "22"
```

I need to remove the comma and convert to numeric to do math:

```
as.numeric(sub(",","",pdbdb$Total))
```

```
[1] 195610  12318  13720   4531    213     22
```

```
#as.numeric(pdbdb$Total)
```

I could turn this inot a function to fix the whole table or any future table I read like this:

```
x <- pdbdb$Total
as.numeric(sub(",","",x))
```

```
[1] 195610  12318  13720   4531    213     22
```

```
comma2numeric <- function(x){
  as.numeric(sub(",","",x))
}
```

```
comma2numeric(pdbdb$X.ray)
```

```
[1] 167192   9639   8730   2869    170     11
```

```
apply(pdbdb, 2, comma2numeric)
```

```
Warning in FUN(newX[, i], ...): NAs introduced by coercion
```

```
     Molecular.Type  X.ray    EM   NMR Multiple.methods Neutron Other  Total
[1,]             NA 167192 15572 12529              208      77    32 195610
[2,]             NA   9639  2635    34                8       2     0  12318
[3,]             NA   8730  4697   286                7       0     0  13720
[4,]             NA   2869   137  1507               14       3     1   4531
[5,]             NA    170    10    33                0       0     0    213
[6,]             NA     11     0     6                1       0     4     22
```

Or try a different read/import function:

```
library(readr)
pdbdb <- read_csv("pdb_stats.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.
```

```
sum(pdbdb$Total)
```

```
[1] 226414
```

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

```
sum(pdbdb$`X-ray`)/sum(pdbdb$Total)*100
```

```
[1] 83.30359
```

x-Ray = 83.30359%

```
sum(pdbdb$`EM`)/sum(pdbdb$Total)*100
```

```
[1] 10.18091
```

EM = 10.18091%

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

```
pdbdb$Total[1]/sum(pdbdb$Total)*100
```

```
[1] 86.39483
```

. 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?

Currently, there are 5 HIV-1 protease structures in the PDB.

Mol*

Mol* (pronounced “molstar”) is a new web-based molecular viewer that we will need to learn the basics of here.

https://molstar.org/viewer/

We will use PDB code: 1HSG

Figure 1: A first image from molstar

Figure 2: The catalytic AP25 amino acid

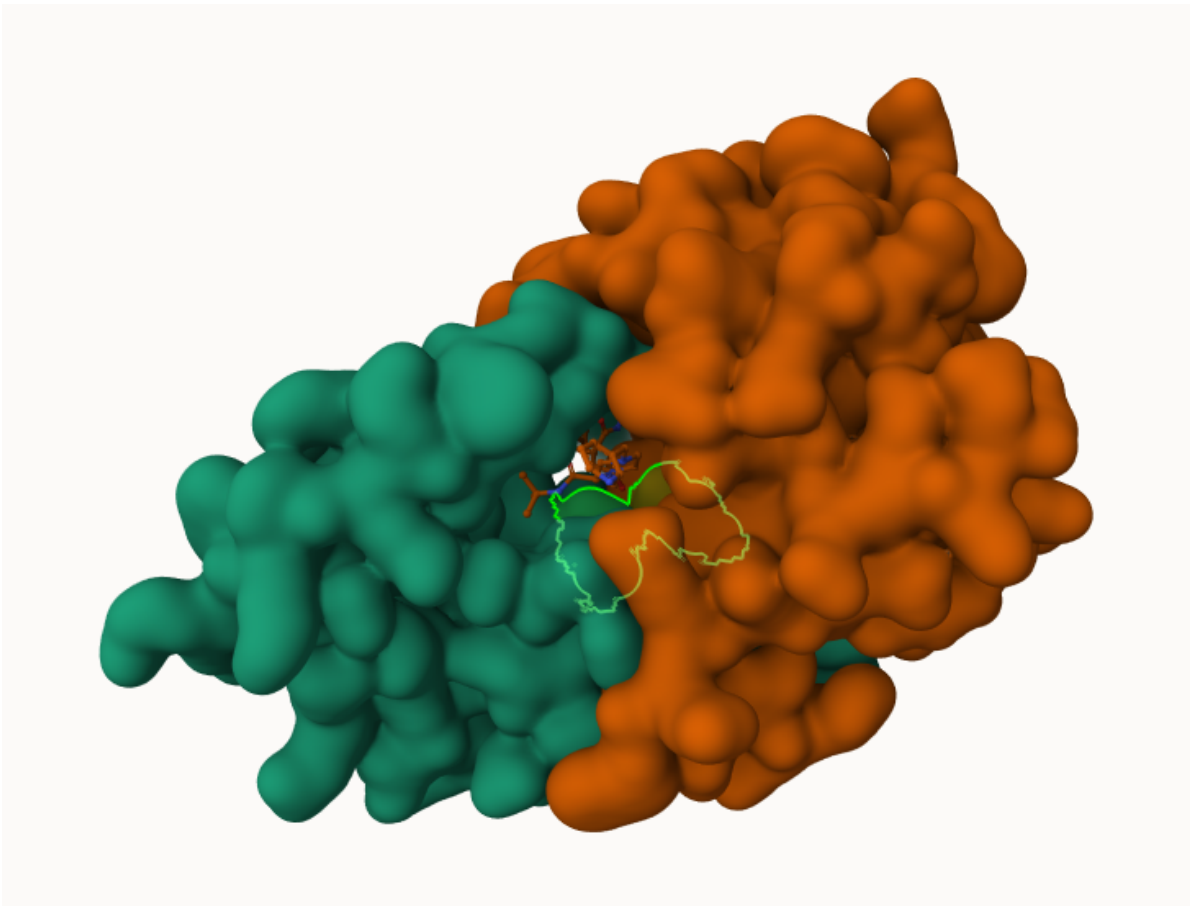


Figure 3: Surface display showing merk compound in the peptide binidng pocket

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

We see just one atom per water molecule because it allows us to focus on the main structural features of the molecule

. 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 has a 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.

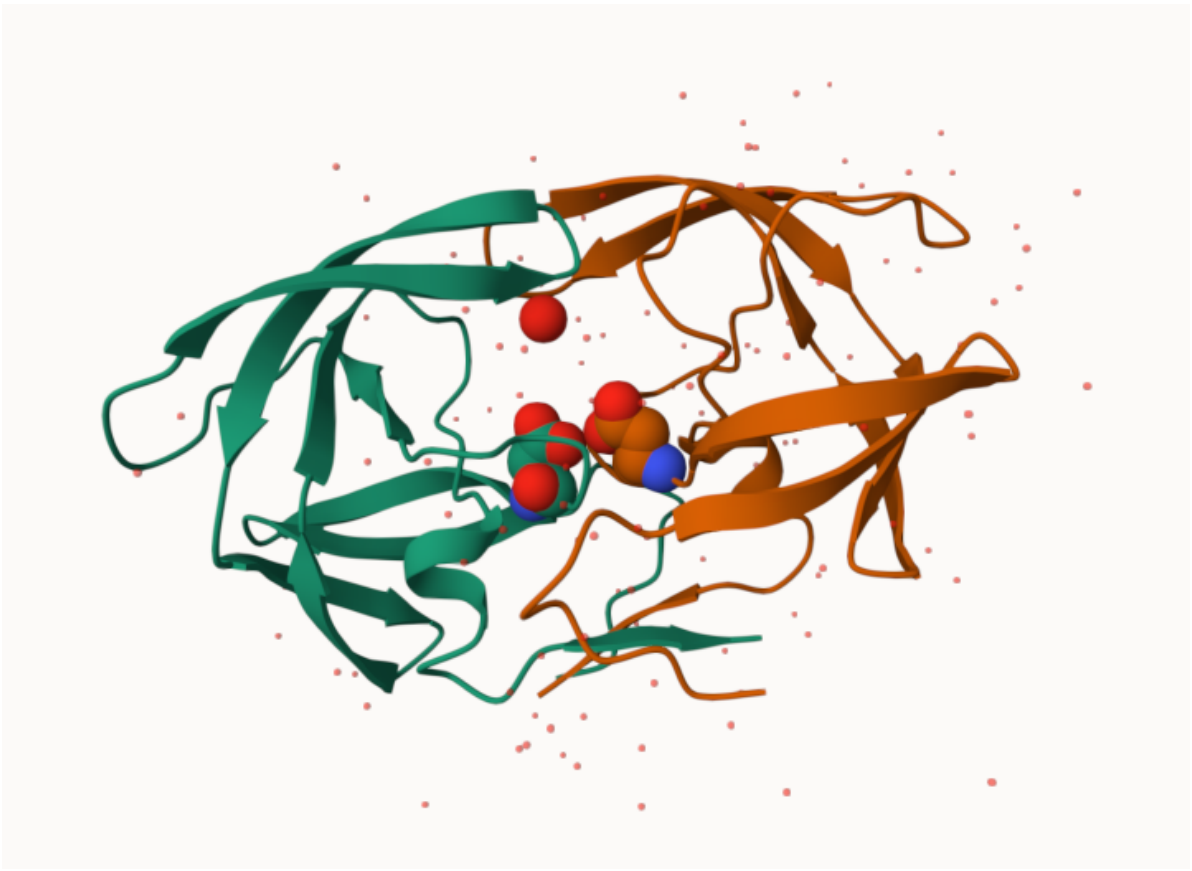


Figure 4: Close up view of binding site with drug and HOH 308

The Bio3D

The bio3d package allows us to do all sorts of structural bioinformations work in R.

Let's start with ohw it can read these PDB files

```
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
```

```
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>
```

```
pdbseq(pdb)
```

```
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
"P" "Q" "I" "T" "L" "W" "Q" "R" "P" "L" "V" "T" "I" "K" "I" "G" "G" "Q" "L" "K"
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
"E" "A" "L" "L" "D" "T" "G" "A" "D" "D" "T" "V" "L" "E" "E" "M" "S" "L" "P" "G"
 41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
"R" "W" "K" "P" "K" "M" "I" "G" "G" "I" "G" "G" "F" "I" "K" "V" "R" "Q" "Y" "D"
 61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
"Q" "I" "L" "I" "E" "I" "C" "G" "H" "K" "A" "I" "G" "T" "V" "L" "V" "G" "P" "T"
 81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99   1
"P" "V" "N" "I" "I" "G" "R" "N" "L" "L" "T" "Q" "I" "G" "C" "T" "L" "N" "F" "P"
  2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21
"Q" "I" "T" "L" "W" "Q" "R" "P" "L" "V" "T" "I" "K" "I" "G" "G" "Q" "L" "K" "E"
 22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41
"A" "L" "L" "D" "T" "G" "A" "D" "D" "T" "V" "L" "E" "E" "M" "S" "L" "P" "G" "R"
 42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61
"W" "K" "P" "K" "M" "I" "G" "G" "I" "G" "G" "F" "I" "K" "V" "R" "Q" "Y" "D" "Q"
 62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81
"I" "L" "I" "E" "I" "C" "G" "H" "K" "A" "I" "G" "T" "V" "L" "V" "G" "P" "T" "P"
 82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99
"V" "N" "I" "I" "G" "R" "N" "L" "L" "T" "Q" "I" "G" "C" "T" "L" "N" "F"
```

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

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

```
[1] 198
```

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

HOH and MK1

. Q9: How many protein chains are in this structure?

2

```
unique(pdb$atom$chain)
```

```
[1] "A" "B"
```

Prediciting functional motions of a single strucutre

Let's do a bioinformatics prediction of function motions - i.e the movements that one of these molecules needs to make to do its stuff

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

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

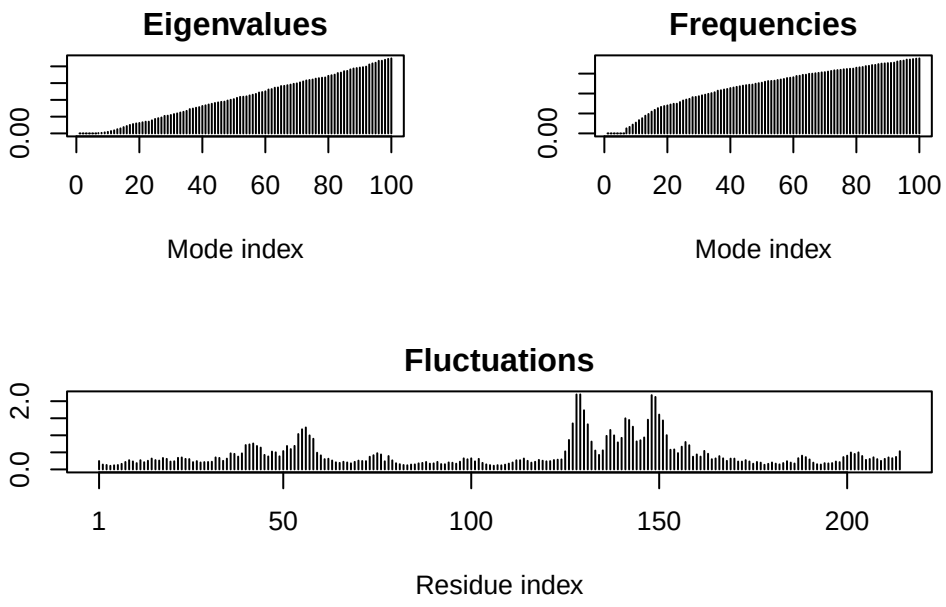
```
adk
```

```
 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:
      MRIILLGAPGAGKGTQAQFIMEKYGIPQISTGDMLRAAVKSGSELGKQAKDIMDAGKLVT
      DELVIALVKERIAQEDCRNGFLLDGFPRTIPQADAMKEAGINVDYVLEFDVPDELIVDKI
      VGRRVHAPSGRVYHVKFNPPKVEGKDDVTGEELTTRKDDQEETVRKRLVEYHQMTAPLIG
      YYSKEAEAGNTKYAKVDGTKPVAEVRADLEKILG

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

```
# Perform flexibility prediction
m <- nma(adk)
```

```
 Building Hessian...        Done in 0.06 seconds.
 Diagonalizing Hessian...   Done in 0.51 seconds.
```

```
plot(m)
```



Write out multi-model PDB file that we can use to make an animation of the predicted motions.

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

Comparative Analysis of protein strucutres

```
library(bio3d)
```

Here we will find and analyze all ADK structures in the PDB database.

We will start with a single database accession id: "1ake_A"

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

```
Warning in get.seq(id): 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
```

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

msa package

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

bio3d

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

True

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

```
ncol(aa$al)
```

```
[1] 214
```

```
attributes(aa)
```

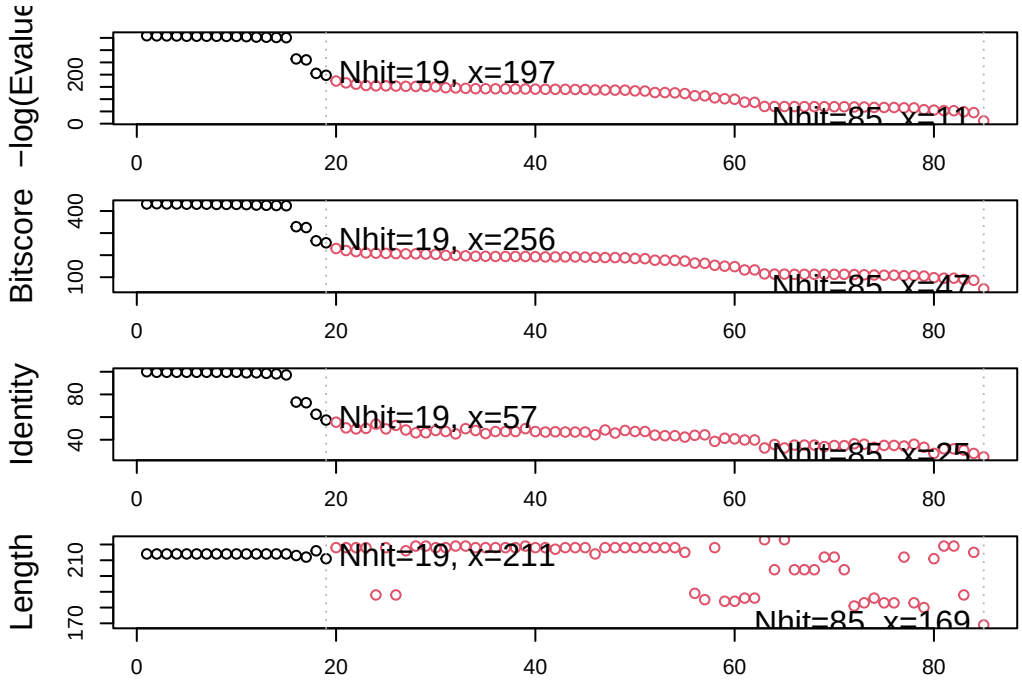
```
$names
[1] "id"   "ali"  "call"

$class
[1] "fasta"
```

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

```
 Searching ... please wait (updates every 5 seconds) RID = JP73N9FY016
 ...
 Reporting 85 hits
```

```
plot(b)
```

```
  * Possible cutoff values:    197 11
            Yielding Nhits:    19 85

  * Chosen cutoff value of:    197
            Yielding Nhits:    19
```




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

```
# 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%
```

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

```
# 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
```

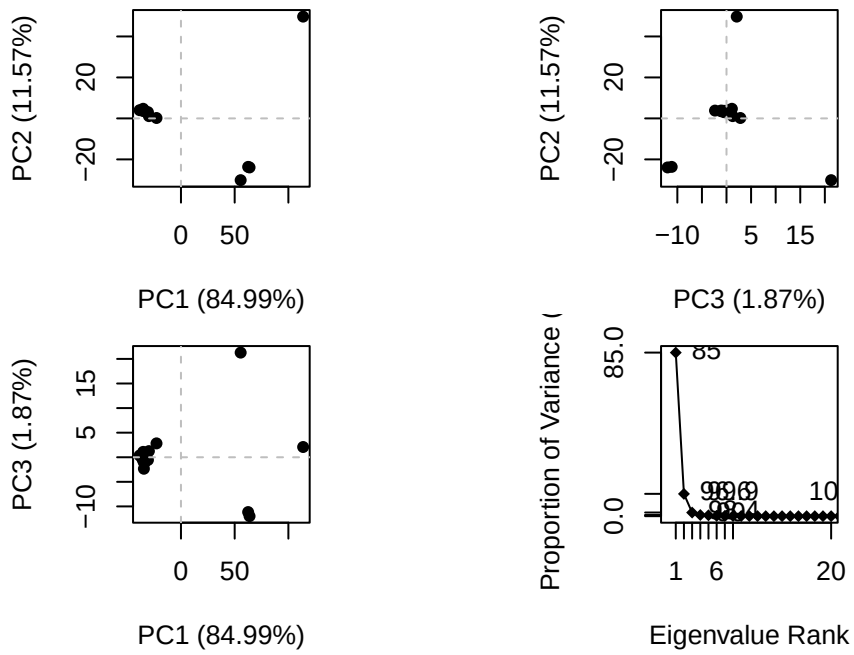
```
pdb
```

```
 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
```

Principle Component Analysis

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



To visualize the major structural variations in the ensemble the function mktrj() can be used to generate a trajectory PDB file by interpolating along a give PC (eigenvector):

```
pc1 <- mktrj(pc.xray, pc=1, file="pc_1.pdb")
```