class 10 bioinformatics

Trinity Lee A16639698

##1: Introduction to the RCSB Protein Data Bank (PDB) The PDB archive is the major repository of information about the 3D structures of large biological molecules, including proteins and nucleic acids. Understanding the shape of these molecules helps to understand how they work. This knowledge can be used to help deduce a structure's role in human health and disease, and in drug development.

downloaded composition stats

```
stats<-read.csv("PDBstats.csv", row.names=1)
stats
```

	X.ray	EM	NMR	Multiple.methods	Neutron	Other
Protein (only)	158,844	11,759	12,296	197	73	32
Protein/Oligosaccharide	9,260	2,054	34	8	1	0
Protein/NA	8,307	3,667	284	7	0	0
Nucleic acid (only)	2,730	113	1,467	13	3	1
Other	164	9	32	0	0	0
Oligosaccharide (only)	11	0	6	1	0	4

	Total
Protein (only)	183,201
Protein/Oligosaccharide	11,357
Protein/NA	12,265
Nucleic acid (only)	4,327
Other	205
Oligosaccharide (only)	22

There is a problem here due to the commas in the numbers. This causes R to treat them as characters.

```
x<-stats$X.ray
x
```

```
[1] "158,844" "9,260"   "8,307"   "2,730"   "164"     "11"
```

```
as.numeric(gsub(",","",x))
```

```
[1] 158844   9260   8307   2730    164     11
```

```
rm.comma<-function(x){as.numeric(gsub(",","", x))}
rm.comma(stats$EM)
```

```
[1] 11759  2054  3667   113     9     0
```

I can use `apply()` to fix the whole table

```
pdbstats<-apply(stats,2,rm.comma)
rownames(pdbstats)<- rownames(stats)
head(pdbstats)
```

```
                         X.ray    EM   NMR Multiple.methods Neutron Other
Protein (only)          158844 11759 12296               197      73    32
Protein/Oligosaccharide   9260  2054    34                 8       1     0
Protein/NA                8307  3667   284                 7       0     0
Nucleic acid (only)       2730   113  1467                13       3     1
Other                      164     9    32                 0       0     0
Oligosaccharide (only)      11     0     6                 1       0     4
                         Total
Protein (only)          183201
Protein/Oligosaccharide  11357
Protein/NA               12265
Nucleic acid (only)       4327
Other                      205
Oligosaccharide (only)      22
```

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

```
totals<-apply(pdbstats,2,sum)
round(totals/totals["Total"]*100,2)
```

```
     X.ray         EM        NMR Multiple.methods
     84.83       8.33       6.68             0.11
   Neutron      Other      Total
      0.04       0.02     100.00
```

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

```
round(pdbstats[1,"Total"]/sum(pdbstats[,"Total"])*100,2)
```

```
[1] 86.67
```

```
round(pdbstats[1,"Total"]/251600768*100,2)
```

```
[1] 0.07
```

Skip Q3

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

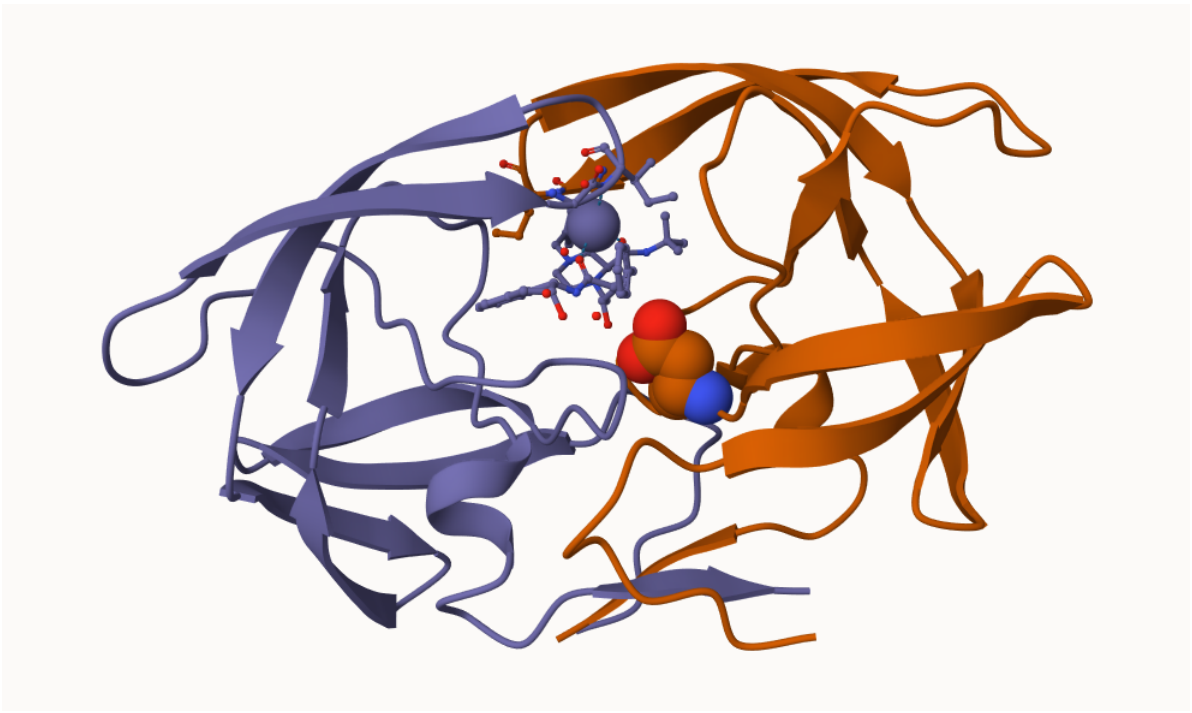
The hydrogen molecules are too small to view. This is a 2 angstrom structure and hydrogen is not visible at this resolution. You need 1 angstrom.

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

Water HOH 308

Q6Q6: 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.

here is a figure of the HIP-Pr with the catalytic ASP residues the MK1 conpound and the all important water 308.



The bio3d package for structural bioinformatics

```
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
```

```
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>
```

##Predicting functional motions of a single structure let's finish today with a bioinformatics calculation to predict the functional motions of a PDB structure.

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

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

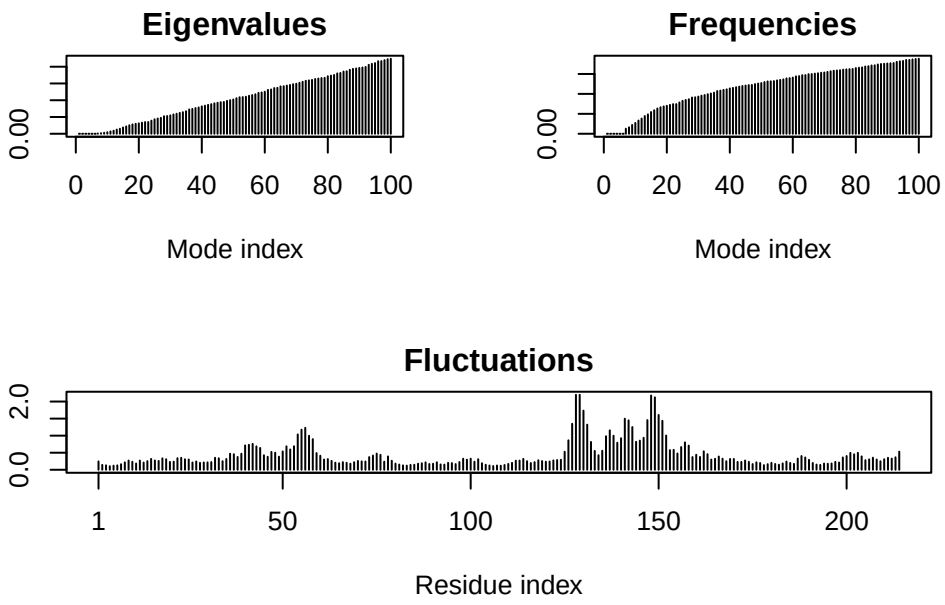
```
adk
```

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

```
 Building Hessian...        Done in 0.031 seconds.
 Diagonalizing Hessian...   Done in 0.406 seconds.
```

```
plot(m)
```




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

##4. Comparative structure analysis of Adenylate Kinase The bio3d package `pca()` function provides a convenient interface for performing PCA of biomolecular structure data.

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.

We will begin by first installing the packages we need for today's session. The `msa()` oackage is from BioConductor. These packages focus on genomics type work and are managed by the `BiocManager` package first Install `BiocManager` and `BiocManager::install("msa")` entered in R console

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

```
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

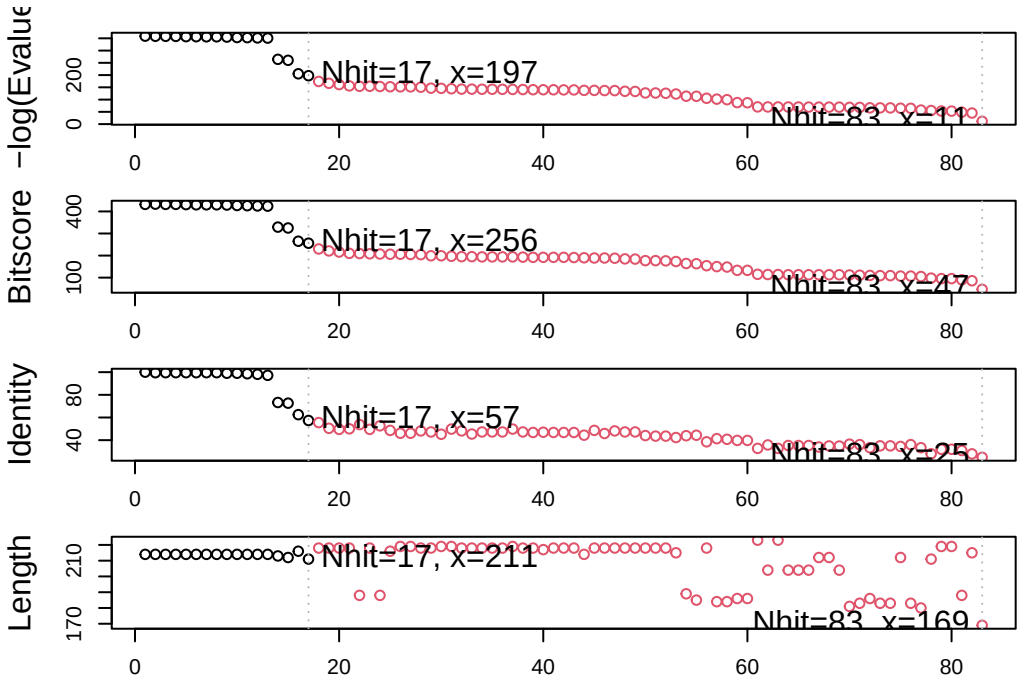
Now I can search the PDB database for related sequences:

```
  #blast or hmmer search
  b<-blast.pdb(aa)
```

```
Searching ... please wait (updates every 5 seconds) RID = MP5F6WJK016
.
Reporting 83 hits
```

```
# Plot a summary of search results
hits<-plot(b)
```

```
 * Possible cutoff values:    197 11
          Yielding Nhits:    17 83

 * Chosen cutoff value of:    197
          Yielding Nhits:    17
```




```
attributes(b)
```

```
$names
[1] "hit.tbl" "raw"     "url"

$class
[1] "blast"
```

```
head(b$hits.tbl)
```

```
NULL
```

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

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

side-note:lets annotate these structures (in other words find out what they are, what species they are from, stuff about the experiments they were solved in etc.)

For this we can use the `pdb.annotate()`

```
anno<-pdb.annotate(hits$pdb.id)
```

```
#attributes(anno)
head(anno)
```

```
       structureId chainId macromoleculeType chainLength experimentalTechnique
1AKE_A        1AKE       A           Protein         214                 X-ray
6S36_A        6S36       A           Protein         214                 X-ray
6RZE_A        6RZE       A           Protein         214                 X-ray
3HPR_A        3HPR       A           Protein         214                 X-ray
1E4V_A        1E4V       A           Protein         214                 X-ray
5EJE_A        5EJE       A           Protein         214                 X-ray
       resolution       scopDomain                   pfam         ligandId
1AKE_A       2.00 Adenylate kinase Adenylate kinase (ADK)              AP5
6S36_A       1.60             <NA> Adenylate kinase (ADK) CL (3),NA,MG (2)
6RZE_A       1.69             <NA> Adenylate kinase (ADK)    NA (3),CL (2)
3HPR_A       2.00             <NA> Adenylate kinase (ADK)              AP5
1E4V_A       1.85 Adenylate kinase Adenylate kinase (ADK)              AP5
5EJE_A       1.90             <NA> Adenylate kinase (ADK)           AP5,CO
                                                ligandName
1AKE_A                  BIS(ADENOSINE)-5'-PENTAPHOSPHATE
6S36_A    CHLORIDE ION (3),SODIUM ION,MAGNESIUM ION (2)
6RZE_A                   SODIUM ION (3),CHLORIDE ION (2)
```

```
3HPR_A                 BIS(ADENOSINE)-5'-PENTAPHOSPHATE
1E4V_A                 BIS(ADENOSINE)-5'-PENTAPHOSPHATE
5EJE_A BIS(ADENOSINE)-5'-PENTAPHOSPHATE,COBALT (II) ION
                                      source
1AKE_A                       Escherichia coli
6S36_A                       Escherichia coli
6RZE_A                       Escherichia coli
3HPR_A                  Escherichia coli K-12
1E4V_A                       Escherichia coli
5EJE_A Escherichia coli O139:H28 str. E24377A

1AKE_A STRUCTURE OF THE COMPLEX BETWEEN ADENYLATE KINASE FROM ESCHERICHIA COLI AND THE INHIBI
6S36_A
6RZE_A
3HPR_A
1E4V_A
5EJE_A                                                                                  Cryst
                                                    citation rObserved  rFree
1AKE_A               Muller, C.W., et al. J Mol Biol (1992)    0.1960     NA
6S36_A                Rogne, P., et al. Biochemistry (2019)    0.1632 0.2356
6RZE_A                Rogne, P., et al. Biochemistry (2019)    0.1865 0.2350
3HPR_A Schrank, T.P., et al. Proc Natl Acad Sci U S A (2009)    0.2100 0.2432
1E4V_A                 Muller, C.W., et al. Proteins (1993)    0.1960     NA
5EJE_A Kovermann, M., et al. Proc Natl Acad Sci U S A (2017)    0.1889 0.2358
        rWork spaceGroup
1AKE_A 0.1960  P 21 2 21
6S36_A 0.1594    C 1 2 1
6RZE_A 0.1819    C 1 2 1
3HPR_A 0.2062  P 21 21 2
1E4V_A 0.1960  P 21 2 21
5EJE_A 0.1863  P 21 2 21
```

now we can download all these structures for further analysis with the `get.pdb()` function

```
# 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


  |
  |                                                                      |   0%
  |
  |=====                                                                 |   8%
  |
```

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

Now we have all these related structures we can align and supperpose

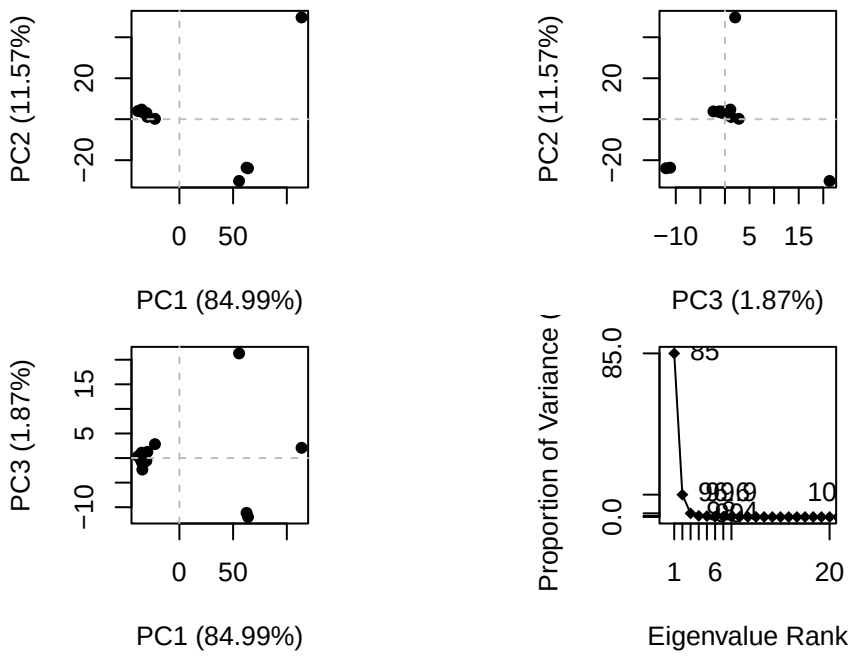
```
# Align releated PDBs
pdbs <- pdbaln(files, fit = TRUE, exefile="msa")
```

```
Reading PDB files:
pdbs/split_chain/1AKE_A.pdb
pdbs/split_chain/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
```

##Principal component analysis

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



##5. Optional further visualization

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

##8. Custom analysis of resulting models

```
results_dir <- "hivpr_dimer_23119"
# File names for all PDB models
pdb_files <- list.files(path=results_dir,
                        pattern="*.pdb",
                        full.names = TRUE)
pdb_files
```

```
[1] "hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_001_alphafold2_multimer_v3_model_1_se
[2] "hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_002_alphafold2_multimer_v3_model_5_se
[3] "hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_003_alphafold2_multimer_v3_model_4_se
[4] "hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_004_alphafold2_multimer_v3_model_3_se
[5] "hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_005_alphafold2_multimer_v3_model_2_se
```

```
library(bio3d)

# Read all data from Models
```

```
#  and superpose/fit coords
pdbs <- pdbaln(pdb_files, fit=TRUE, exefile="msa")
```

```
Reading PDB files:
hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_001_alphafold2_multimer_v3_model_1_seed_0
hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_002_alphafold2_multimer_v3_model_5_seed_0
hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_003_alphafold2_multimer_v3_model_4_seed_0
hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_004_alphafold2_multimer_v3_model_3_seed_0
hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_005_alphafold2_multimer_v3_model_2_seed_0
.....

Extracting sequences

pdb/seq: 1   name: hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_001_alphafold2_multimer
pdb/seq: 2   name: hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_002_alphafold2_multimer
pdb/seq: 3   name: hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_003_alphafold2_multimer
pdb/seq: 4   name: hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_004_alphafold2_multimer
pdb/seq: 5   name: hivpr_dimer_23119/hivpr_dimer_23119_unrelaxed_rank_005_alphafold2_multimer
```

```
pdbs
```

```
                                 1        .        .        .        .        50
[Truncated_Name:1]hivpr_dime   PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:2]hivpr_dime   PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:3]hivpr_dime   PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:4]hivpr_dime   PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:5]hivpr_dime   PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
                               **************************************************
                                 1        .        .        .        .        50

                                51        .        .        .        .        100
[Truncated_Name:1]hivpr_dime   GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:2]hivpr_dime   GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:3]hivpr_dime   GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:4]hivpr_dime   GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:5]hivpr_dime   GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
                               **************************************************
                                51        .        .        .        .        100

                               101        .        .        .        .        150
```

```
[Truncated_Name:1]hivpr_dime   QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIG
[Truncated_Name:2]hivpr_dime   QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIG
[Truncated_Name:3]hivpr_dime   QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIG
[Truncated_Name:4]hivpr_dime   QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIG
[Truncated_Name:5]hivpr_dime   QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIG
                               **************************************************
                             101        .         .         .         .         150

                             151        .         .         .         .       198
[Truncated_Name:1]hivpr_dime   GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:2]hivpr_dime   GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:3]hivpr_dime   GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:4]hivpr_dime   GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:5]hivpr_dime   GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
                               **********************************************
                             151        .         .         .         .       198

Call:
  pdbaln(files = pdb_files, fit = TRUE, exefile = "msa")

Class:
  pdbs, fasta

Alignment dimensions:
  5 sequence rows; 198 position columns (198 non-gap, 0 gap)

+ attr: xyz, resno, b, chain, id, ali, resid, sse, call
```

Calculate the RMSD between all models.

```
rd <- rmsd(pdbs)
```

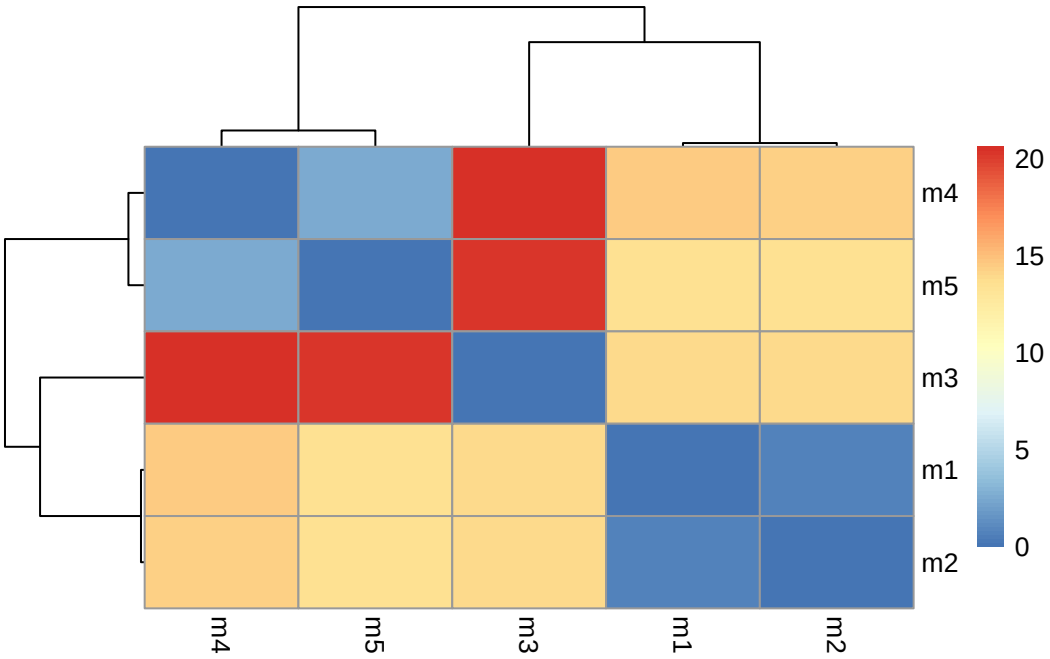
```
Warning in rmsd(pdbs): No indices provided, using the 198 non NA positions
```

```
range(rd)
```

```
[1]  0.000 20.591
```

Draw a heatmap of RMSD matrix values

```
library(pheatmap)

colnames(rd) <- paste0("m",1:5)
rownames(rd) <- paste0("m",1:5)
pheatmap(rd)
```



And a plot pLDDT values across all models

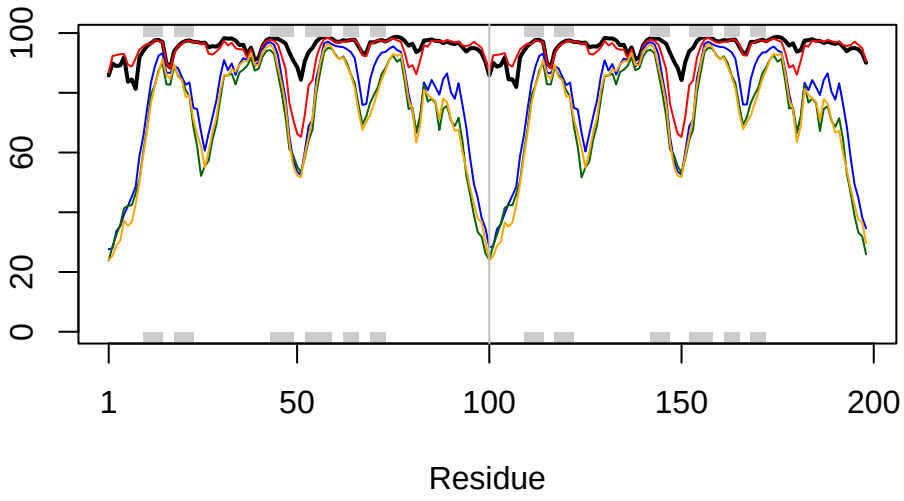
```
# Read a reference PDB structure
pdb <- read.pdb("1hsg")
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/vx/w7ph64q100qd2g769hxkrnrw0000gn/T//RtmphW4kPP/1hsg.pdb exists.
Skipping download
```

You could optionally obtain secondary structure from a call to stride() or dssp() on any of the model structures.

```
plotb3(pdbs$b, typ="l", lwd=2, sse=pdb)
points(pdbs$b[2,], typ="l", col="red")
points(pdbs$b[3,], typ="l", col="blue")
points(pdbs$b[4,], typ="l", col="darkgreen")
points(pdbs$b[5,], typ="l", col="orange")
abline(v=100, col="gray")
```



We can improve the superposition/fitting of our models by finding the most consistent "rigid core" common across all the models. For this we will use the core.find() function:

```
core <- core.find(pdbs)
```

```
core size 197 of 198  vol = 4696.464
core size 196 of 198  vol = 3949.046
core size 195 of 198  vol = 3694.692
core size 194 of 198  vol = 3464.819
core size 193 of 198  vol = 3284.063
core size 192 of 198  vol = 3080.418
core size 191 of 198  vol = 2922.025
core size 190 of 198  vol = 2799.382
core size 189 of 198  vol = 2728.707
```

```
core size 188 of 198  vol = 2671.239
core size 187 of 198  vol = 2637.085
core size 186 of 198  vol = 2619.983
core size 185 of 198  vol = 2667.449
core size 184 of 198  vol = 2736.952
core size 183 of 198  vol = 2855.679
core size 182 of 198  vol = 3000.642
core size 181 of 198  vol = 3112.274
core size 180 of 198  vol = 3195.289
core size 179 of 198  vol = 3231.995
core size 178 of 198  vol = 3271.691
core size 177 of 198  vol = 3279.323
core size 176 of 198  vol = 3245.561
core size 175 of 198  vol = 3202.913
core size 174 of 198  vol = 3104.668
core size 173 of 198  vol = 2995.483
core size 172 of 198  vol = 2882.964
core size 171 of 198  vol = 2781.012
core size 170 of 198  vol = 2708.845
core size 169 of 198  vol = 2623.492
core size 168 of 198  vol = 2550.511
core size 167 of 198  vol = 2473.008
core size 166 of 198  vol = 2403.471
core size 165 of 198  vol = 2327.791
core size 164 of 198  vol = 2230.613
core size 163 of 198  vol = 2136.51
core size 162 of 198  vol = 2072.598
core size 161 of 198  vol = 1988.682
core size 160 of 198  vol = 1911.32
core size 159 of 198  vol = 1852.596
core size 158 of 198  vol = 1776.26
core size 157 of 198  vol = 1711.271
core size 156 of 198  vol = 1645.377
core size 155 of 198  vol = 1591.953
core size 154 of 198  vol = 1523.012
core size 153 of 198  vol = 1462.423
core size 152 of 198  vol = 1412.06
core size 151 of 198  vol = 1344.849
core size 150 of 198  vol = 1281.267
core size 149 of 198  vol = 1235.254
core size 148 of 198  vol = 1175.997
core size 147 of 198  vol = 1130.458
core size 146 of 198  vol = 1088.097
```

```
core size 145 of 198  vol = 1050.266
core size 144 of 198  vol = 1001.324
core size 143 of 198  vol = 956.163
core size 142 of 198  vol = 906.737
core size 141 of 198  vol = 871.969
core size 140 of 198  vol = 841.445
core size 139 of 198  vol = 808.348
core size 138 of 198  vol = 770.142
core size 137 of 198  vol = 727.97
core size 136 of 198  vol = 685.892
core size 135 of 198  vol = 648.926
core size 134 of 198  vol = 619.519
core size 133 of 198  vol = 589.972
core size 132 of 198  vol = 558.979
core size 131 of 198  vol = 532.435
core size 130 of 198  vol = 511.903
core size 129 of 198  vol = 492.416
core size 128 of 198  vol = 467.492
core size 127 of 198  vol = 441.412
core size 126 of 198  vol = 409.527
core size 125 of 198  vol = 384.007
core size 124 of 198  vol = 363.926
core size 123 of 198  vol = 350.246
core size 122 of 198  vol = 325.486
core size 121 of 198  vol = 298.625
core size 120 of 198  vol = 278.072
core size 119 of 198  vol = 256.676
core size 118 of 198  vol = 243.447
core size 117 of 198  vol = 228.965
core size 116 of 198  vol = 220.708
core size 115 of 198  vol = 209.913
core size 114 of 198  vol = 199.129
core size 113 of 198  vol = 182.235
core size 112 of 198  vol = 165.009
core size 111 of 198  vol = 150.273
core size 110 of 198  vol = 139.388
core size 109 of 198  vol = 126.387
core size 108 of 198  vol = 115.355
core size 107 of 198  vol = 107.148
core size 106 of 198  vol = 99.636
core size 105 of 198  vol = 93.061
core size 104 of 198  vol = 85.839
core size 103 of 198  vol = 77.556
```

```
 core size 102 of 198  vol = 70.699
 core size 101 of 198  vol = 65.538
 core size 100 of 198  vol = 60.254
 core size 99 of 198  vol = 55.144
 core size 98 of 198  vol = 50.144
 core size 97 of 198  vol = 45.734
 core size 96 of 198  vol = 40.319
 core size 95 of 198  vol = 33.261
 core size 94 of 198  vol = 25.746
 core size 93 of 198  vol = 18.883
 core size 92 of 198  vol = 13.586
 core size 91 of 198  vol = 7.87
 core size 90 of 198  vol = 5.132
 core size 89 of 198  vol = 3.326
 core size 88 of 198  vol = 2.358
 core size 87 of 198  vol = 1.835
 core size 86 of 198  vol = 1.518
 core size 85 of 198  vol = 1.29
 core size 84 of 198  vol = 1.109
 core size 83 of 198  vol = 0.917
 core size 82 of 198  vol = 0.802
 core size 81 of 198  vol = 0.679
 core size 80 of 198  vol = 0.576
 core size 79 of 198  vol = 0.524
 core size 78 of 198  vol = 0.479
 FINISHED: Min vol ( 0.5 ) reached
```

```
core.inds <- print(core, vol=0.5)
```

```
# 79 positions (cumulative volume <= 0.5 Angstrom^3)
  start end length
1    10  25     16
2    27  48     22
3    53  93     41
```

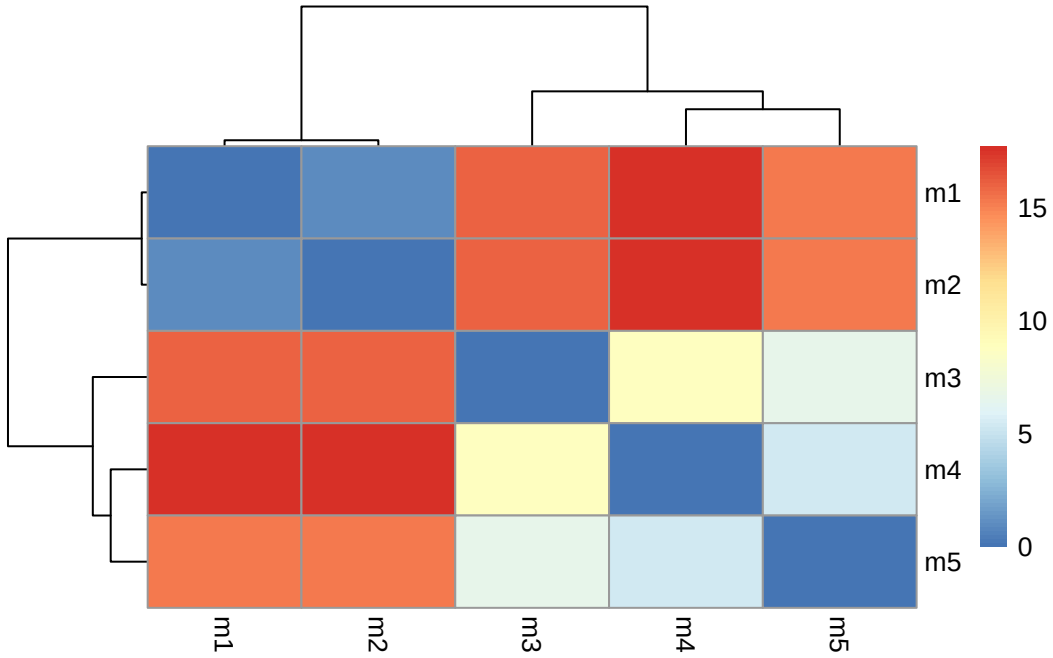
```
xyz <- pdbfit(pdbs, core.inds, outpath="corefit_structures")
```

```
rd <- rmsd(xyz)
```

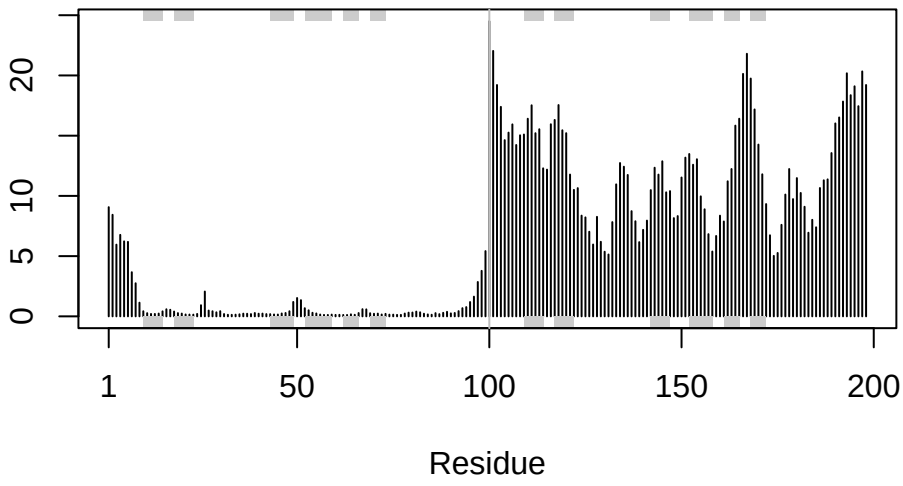
```
Warning in rmsd(xyz): No indices provided, using the 198 non NA positions
```

```
# Change the names for easy reference
colnames(rd) <- paste0("m",1:5)
rownames(rd) <- paste0("m",1:5)
```

```
pheatmap(rd)
```




```
rf <- rmsf(xyz)

plotb3(rf, sse=pdb)
abline(v=100, col="gray", ylab="RMSF")
```



##Predicted Alignment Error for domains

```
library(jsonlite)

# Listing of all PAE JSON files
pae_files <- list.files(path=results_dir,
                        pattern=".*model.*\\.json",
                        full.names = TRUE)
#For example purposes lets read the 1st and 5th files

pae1 <- read_json(pae_files[1],simplifyVector = TRUE)
pae5 <- read_json(pae_files[5],simplifyVector = TRUE)

attributes(pae1)
```

```
$names
[1] "plddt"   "max_pae" "pae"     "ptm"     "iptm"
```

```
# Per-residue pLDDT scores
#  same as B-factor of PDB..
head(pae1$plddt)
```

```
[1] 85.81 89.81 88.94 89.19 91.94 83.69
```

```
pae1$max_pae
```

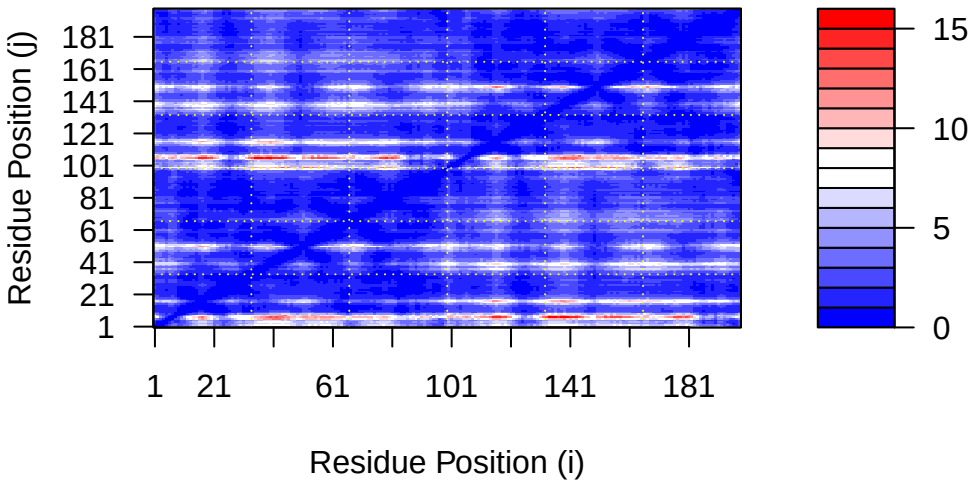
```
[1] 15.83594
```

```
pae5$max_pae
```

```
[1] 29.23438
```

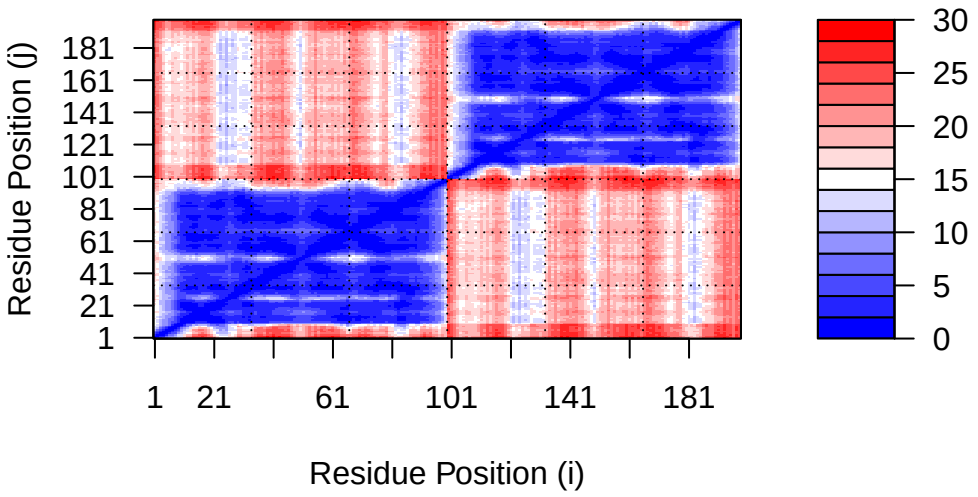
We can plot these with ggplot or with functions from the Bio3D package:

```
plot.dmat(pae1$pae,
          xlab="Residue Position (i)",
          ylab="Residue Position (j)")
```



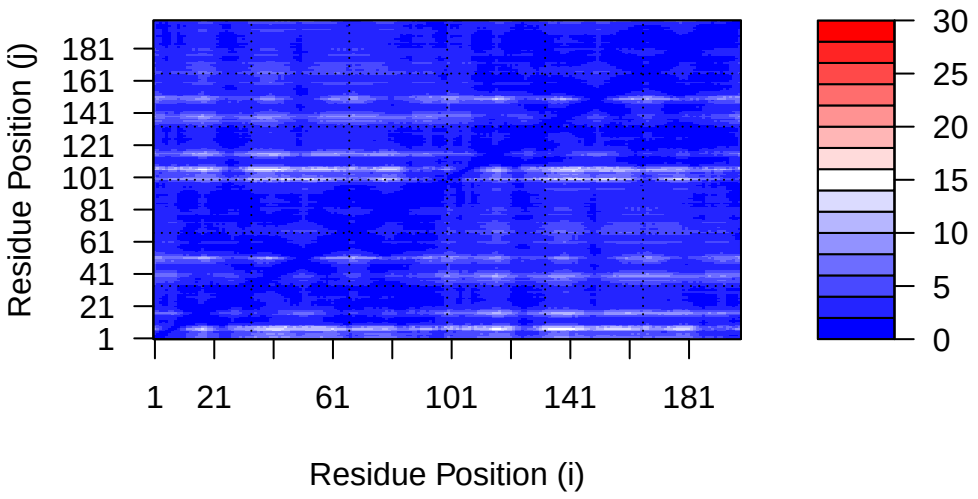

```
plot.dmat(pae5$pae,
          xlab="Residue Position (i)",
```

```
          ylab="Residue Position (j)",
          grid.col = "black",
          zlim=c(0,30))
```



#We should really plot all of these using the same z range. Here is the model 1 plot again but this time using the same data range as the plot for model 5:

```
plot.dmat(pae1$pae,
          xlab="Residue Position (i)",
          ylab="Residue Position (j)",
          grid.col = "black",
          zlim=c(0,30))
```



Residue conservation from alignment file

```
aln_file <- list.files(path=results_dir,
                       pattern=".a3m$",
                        full.names = TRUE)
aln_file
```

```
[1] "hivpr_dimer_23119/hivpr_dimer_23119.a3m"
```

```
aln <- read.fasta(aln_file[1], to.upper = TRUE)
```

```
[1] " ** Duplicated sequence id's: 101 **"
[2] " ** Duplicated sequence id's: 101 **"
```

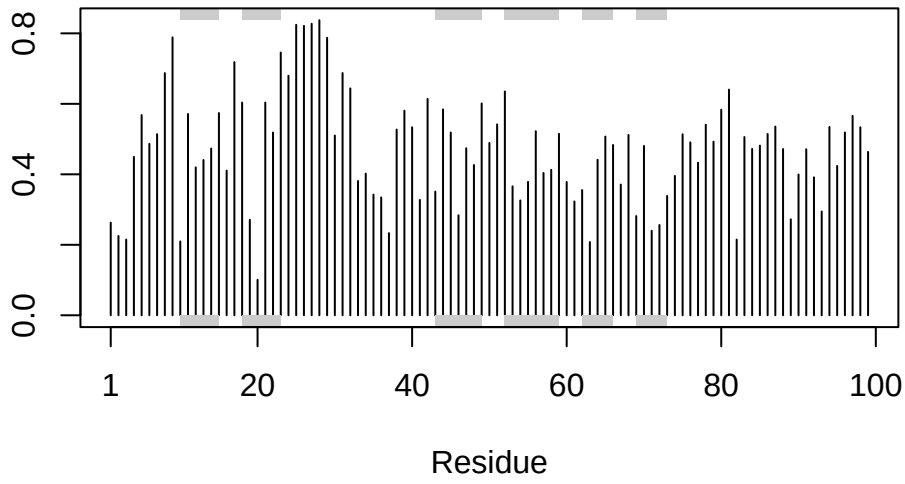
How many sequences are in this alignment

```
dim(aln$ali)
```

```
[1] 5378  132
```

We can score residue conservation in the alignment with the `conserv()` function.

```
sim <- conserv(aln)

plotb3(sim[1:99], sse=trim.pdb(pdb, chain="A"))
```




```
con <- consensus(aln, cutoff = 0.9)
con$seq
```

```
  [1] "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-"
 [19] "-" "-" "-" "-" "-" "-" "D" "T" "G" "A" "-" "-" "-" "-" "-" "-" "-" "-"
 [37] "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-"
 [55] "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-"
 [73] "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-"
 [91] "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-"
[109] "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "-"
[127] "-" "-" "-" "-" "-" "-"
```

For a final visualization we can map this conservation score to the Occupancy column of a PDB file for viewing in molecular viewer programs such as Mol*, PyMol, VMD, chimera etc.

```
m1.pdb <- read.pdb(pdb_files[1])
occ <- vec2resno(c(sim[1:99], sim[1:99]), m1.pdb$atom$resno)
write.pdb(m1.pdb, o=occ, file="m1_conserv.pdb")
```