Class 11: Structural Bioinformatics Part 2

AUTHOR
Nundini Vatshney (PID: A16867985)

AlphaFold has changed the game for protein structure prediction and allows anyone with sufficient bioinformatics skills to predict the structure of virtually any protein.

We ran AlphaFold via GoogleColab at: https://github.com/sokrypton/ColabFold

In particular, we used their AlphaFold2_mmseqs2 version that uses mmseqs2 rather than Hmmer for sequence search.

The main outputs include a set of **PDB structure** files along with matching **JSON format files** that tell us how good the resulting models might be.

Let's start by loading these structures up in Mol*

```
library(bio3d)
```

```
results_dir <- "HIVPr1Dimer_23119"
pdb.files <- list.files(path="HIVPr1Dimer_23119",
                        pattern = ".pdb",
                        full.names = TRUE)

basename(pdb.files)
```

```
[1] "HIVPr1Dimer_23119_unrelaxed_rank_001_alphafold2_multimer_v3_model_1_seed_000.pdb"
[2] "HIVPr1Dimer_23119_unrelaxed_rank_002_alphafold2_multimer_v3_model_5_seed_000.pdb"
[3] "HIVPr1Dimer_23119_unrelaxed_rank_003_alphafold2_multimer_v3_model_4_seed_000.pdb"
[4] "HIVPr1Dimer_23119_unrelaxed_rank_004_alphafold2_multimer_v3_model_2_seed_000.pdb"
[5] "HIVPr1Dimer_23119_unrelaxed_rank_005_alphafold2_multimer_v3_model_3_seed_000.pdb"
```

```
# Read all data from Models
#  and superpose/fit coords
pdbs <- pdbaln(pdb.files, fit=TRUE, exefile="msa")
```

```
Reading PDB files:
HIVPr1Dimer_23119/HIVPr1Dimer_23119_unrelaxed_rank_001_alphafold2_multimer_v3_model_1_seed_000.pd
b
HIVPr1Dimer_23119/HIVPr1Dimer_23119_unrelaxed_rank_002_alphafold2_multimer_v3_model_5_seed_000.pd
b
HIVPr1Dimer_23119/HIVPr1Dimer_23119_unrelaxed_rank_003_alphafold2_multimer_v3_model_4_seed_000.pd
b
HIVPr1Dimer_23119/HIVPr1Dimer_23119_unrelaxed_rank_004_alphafold2_multimer_v3_model_2_seed_000.pd
b
HIVPr1Dimer_23119/HIVPr1Dimer_23119_unrelaxed_rank_005_alphafold2_multimer_v3_model_3_seed_000.pd
b
```

```
.....

Extracting sequences

pdb/seq: 1   name:
HIVPr1Dimer_23119/HIVPr1Dimer_23119_unrelaxed_rank_001_alphafold2_multimer_v3_model_1_seed_000.pd
b
pdb/seq: 2   name:
HIVPr1Dimer_23119/HIVPr1Dimer_23119_unrelaxed_rank_002_alphafold2_multimer_v3_model_5_seed_000.pd
b
pdb/seq: 3   name:
HIVPr1Dimer_23119/HIVPr1Dimer_23119_unrelaxed_rank_003_alphafold2_multimer_v3_model_4_seed_000.pd
b
pdb/seq: 4   name:
HIVPr1Dimer_23119/HIVPr1Dimer_23119_unrelaxed_rank_004_alphafold2_multimer_v3_model_2_seed_000.pd
b
pdb/seq: 5   name:
HIVPr1Dimer_23119/HIVPr1Dimer_23119_unrelaxed_rank_005_alphafold2_multimer_v3_model_3_seed_000.pd
b
```

```
rd <- rmsd(pdbs, fit=T)
```

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

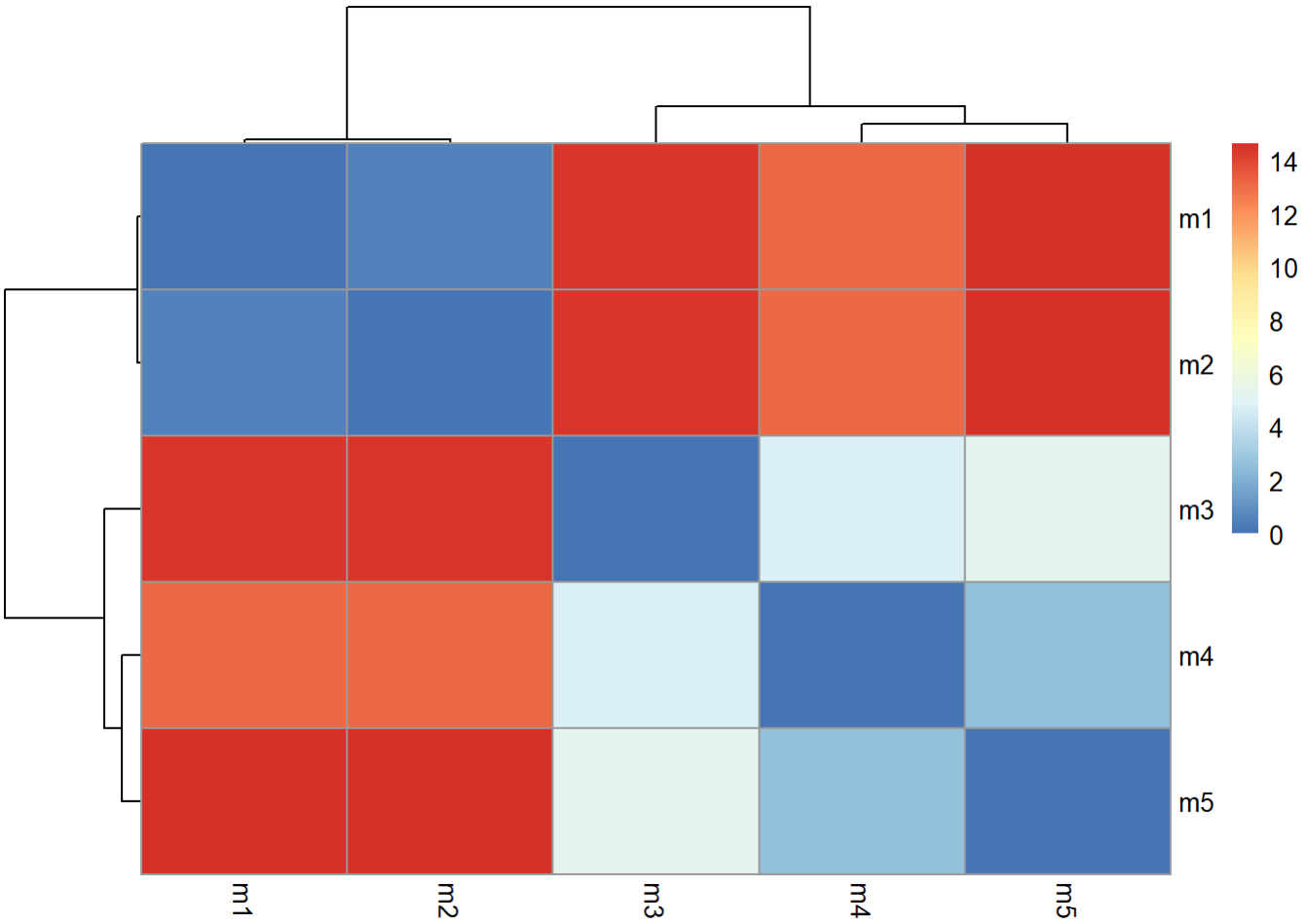
```
range(rd)
```

```
[1]  0.000 14.631
```

Draw a heatmap of these RMSD matrix values

```
library(pheatmap)

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

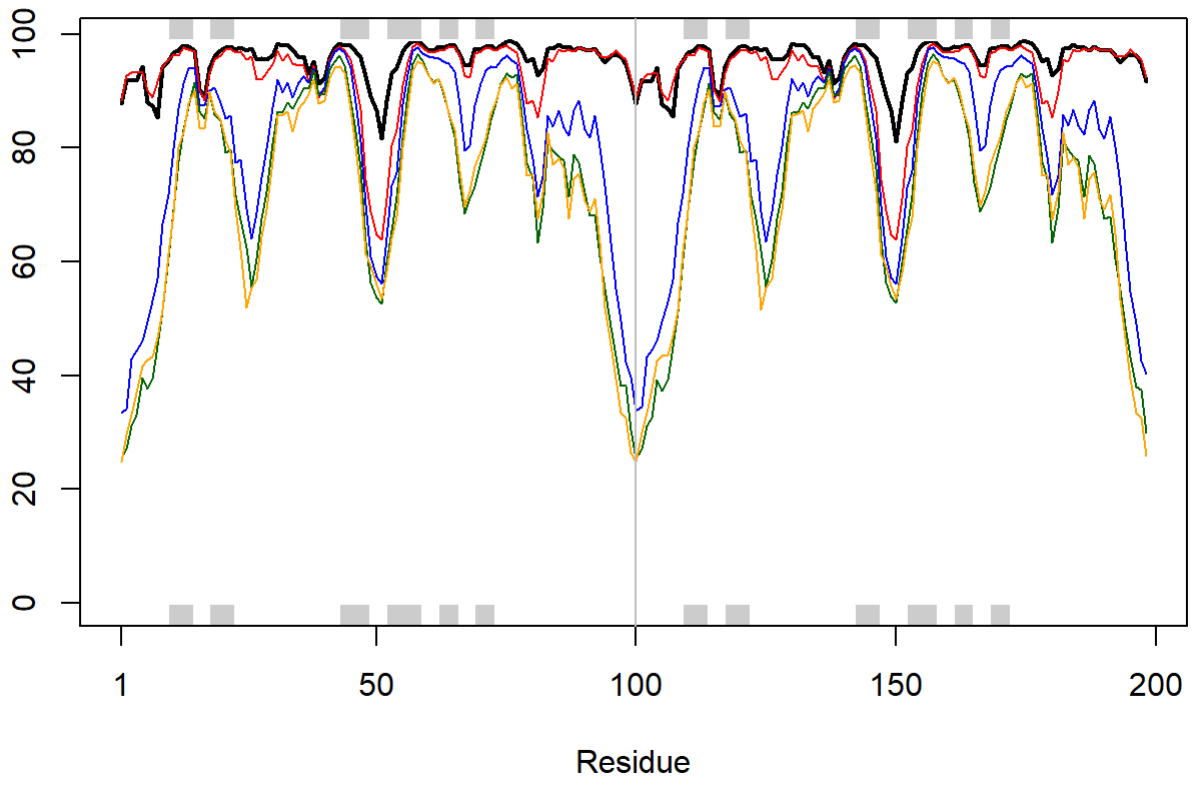


Now lets plot the pLDDT values across all models.

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

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

```
plotb3(pdbs$b[1,], typ="l", lwd=2, sse=pdb)
points(pdbs$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.

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

```
core size 197 of 198  vol = 4578.352
core size 196 of 198  vol = 3931.113
core size 195 of 198  vol = 3709.737
core size 194 of 198  vol = 3496.023
core size 193 of 198  vol = 3302.436
core size 192 of 198  vol = 3146.478
core size 191 of 198  vol = 3048.969
core size 190 of 198  vol = 2970.358
core size 189 of 198  vol = 2893.016
core size 188 of 198  vol = 2831.829
core size 187 of 198  vol = 2774.51
core size 186 of 198  vol = 2728.049
core size 185 of 198  vol = 2704.953
core size 184 of 198  vol = 2701.988
core size 183 of 198  vol = 2715.915
core size 182 of 198  vol = 2809.862
core size 181 of 198  vol = 2888.961
core size 180 of 198  vol = 2967.296
core size 179 of 198  vol = 3036.25
```

```
core size 178 of 198  vol = 3066.28
core size 177 of 198  vol = 3096.829
core size 176 of 198  vol = 3056.412
core size 175 of 198  vol = 3014.767
core size 174 of 198  vol = 2975.012
core size 173 of 198  vol = 2898.05
core size 172 of 198  vol = 2810.173
core size 171 of 198  vol = 2747.532
core size 170 of 198  vol = 2684.434
core size 169 of 198  vol = 2620.353
core size 168 of 198  vol = 2550.877
core size 167 of 198  vol = 2492.582
core size 166 of 198  vol = 2422.978
core size 165 of 198  vol = 2358.916
core size 164 of 198  vol = 2298.292
core size 163 of 198  vol = 2235.918
core size 162 of 198  vol = 2171.02
core size 161 of 198  vol = 2093.559
core size 160 of 198  vol = 2029.144
core size 159 of 198  vol = 1950.957
core size 158 of 198  vol = 1881.015
core size 157 of 198  vol = 1801.506
core size 156 of 198  vol = 1728.892
core size 155 of 198  vol = 1660.037
core size 154 of 198  vol = 1586.149
core size 153 of 198  vol = 1532.718
core size 152 of 198  vol = 1460.186
core size 151 of 198  vol = 1399.251
core size 150 of 198  vol = 1333.908
core size 149 of 198  vol = 1271.747
core size 148 of 198  vol = 1219.496
core size 147 of 198  vol = 1176.003
core size 146 of 198  vol = 1138.478
core size 145 of 198  vol = 1102.124
core size 144 of 198  vol = 1049.642
core size 143 of 198  vol = 1014.063
core size 142 of 198  vol = 970.575
core size 141 of 198  vol = 929.178
core size 140 of 198  vol = 889.104
core size 139 of 198  vol = 846.668
core size 138 of 198  vol = 805.8
core size 137 of 198  vol = 775.034
core size 136 of 198  vol = 743.09
core size 135 of 198  vol = 715.695
core size 134 of 198  vol = 689.788
core size 133 of 198  vol = 660.329
core size 132 of 198  vol = 630.966
core size 131 of 198  vol = 597.207
core size 130 of 198  vol = 566.989
core size 129 of 198  vol = 532.89
core size 128 of 198  vol = 496.208
```

```
core size 127 of 198  vol = 463.183
core size 126 of 198  vol = 431.893
core size 125 of 198  vol = 408.864
core size 124 of 198  vol = 376.61
core size 123 of 198  vol = 362.377
core size 122 of 198  vol = 353.633
core size 121 of 198  vol = 331.501
core size 120 of 198  vol = 312.518
core size 119 of 198  vol = 286.715
core size 118 of 198  vol = 262.336
core size 117 of 198  vol = 245.109
core size 116 of 198  vol = 228.342
core size 115 of 198  vol = 210.366
core size 114 of 198  vol = 197.519
core size 113 of 198  vol = 179.392
core size 112 of 198  vol = 161.891
core size 111 of 198  vol = 148.359
core size 110 of 198  vol = 134.477
core size 109 of 198  vol = 121.261
core size 108 of 198  vol = 109.516
core size 107 of 198  vol = 103.031
core size 106 of 198  vol = 96.443
core size 105 of 198  vol = 88.455
core size 104 of 198  vol = 81.816
core size 103 of 198  vol = 74.88
core size 102 of 198  vol = 68.386
core size 101 of 198  vol = 65.937
core size 100 of 198  vol = 62.345
core size 99 of 198  vol = 58.836
core size 98 of 198  vol = 52.868
core size 97 of 198  vol = 47.796
core size 96 of 198  vol = 41.292
core size 95 of 198  vol = 33.831
core size 94 of 198  vol = 24.912
core size 93 of 198  vol = 18.912
core size 92 of 198  vol = 12.7
core size 91 of 198  vol = 7.35
core size 90 of 198  vol = 4.922
core size 89 of 198  vol = 3.421
core size 88 of 198  vol = 2.553
core size 87 of 198  vol = 1.917
core size 86 of 198  vol = 1.513
core size 85 of 198  vol = 1.201
core size 84 of 198  vol = 1.046
core size 83 of 198  vol = 0.922
core size 82 of 198  vol = 0.755
core size 81 of 198  vol = 0.668
core size 80 of 198  vol = 0.596
core size 79 of 198  vol = 0.549
```

```
core size 78 of 198  vol = 0.493
FINISHED: Min vol ( 0.5 ) reached
```

```
#We can now use the identified core atom positions as a basis for a more suitable superposition a

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

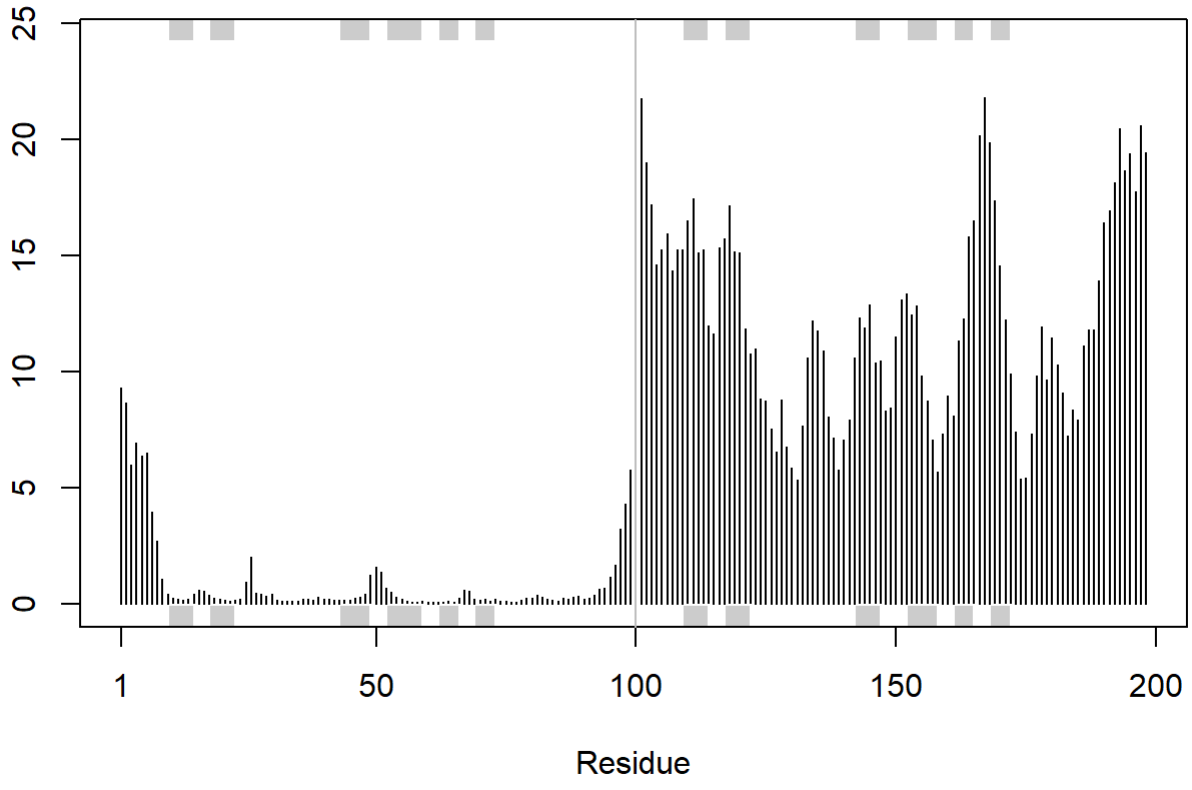
```
# 79 positions (cumulative volume <= 0.5 Angstrom^3)
  start end length
1    10  25     16
2    28  48     21
3    53  94     42
```

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

Now we can examine the RMSF between positions of the structure. RMSF is an often used measure of conformational variance along the structure:

```
rf <- rmsf(xyz)

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



```
#pdb$atom
```

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)
```

```
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] 87.81 92.00 91.81 91.88 94.25 88.00
```

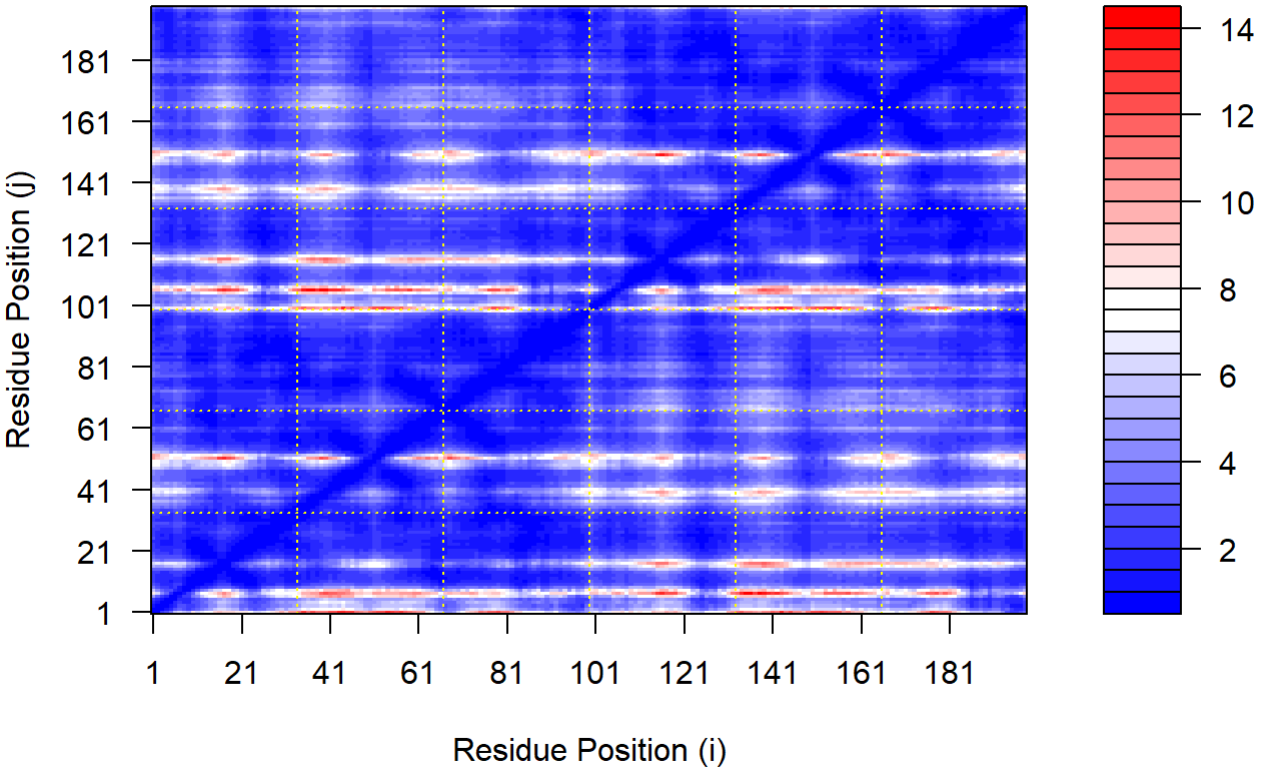
```
pae1$max_pae
```

```
[1] 14.09375
```

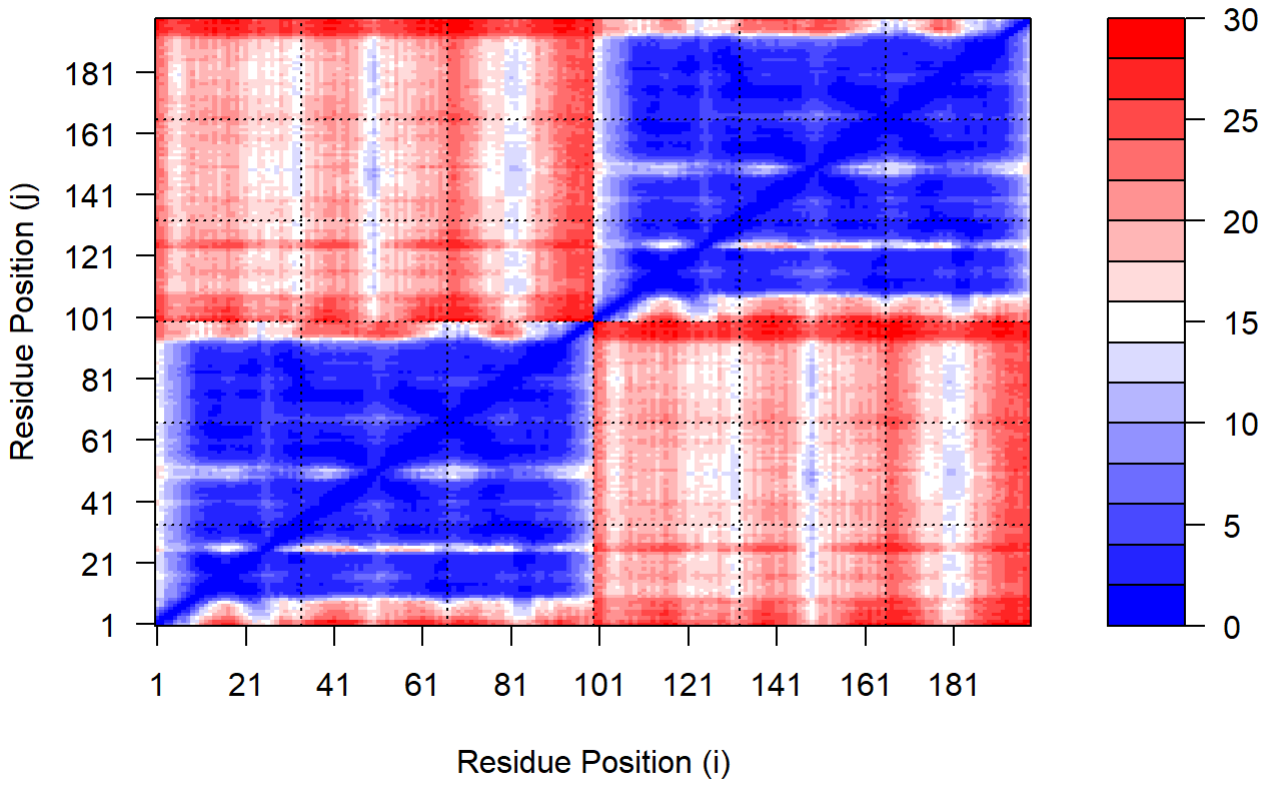
```
pae5$max_pae
```

```
[1] 29.29688
```

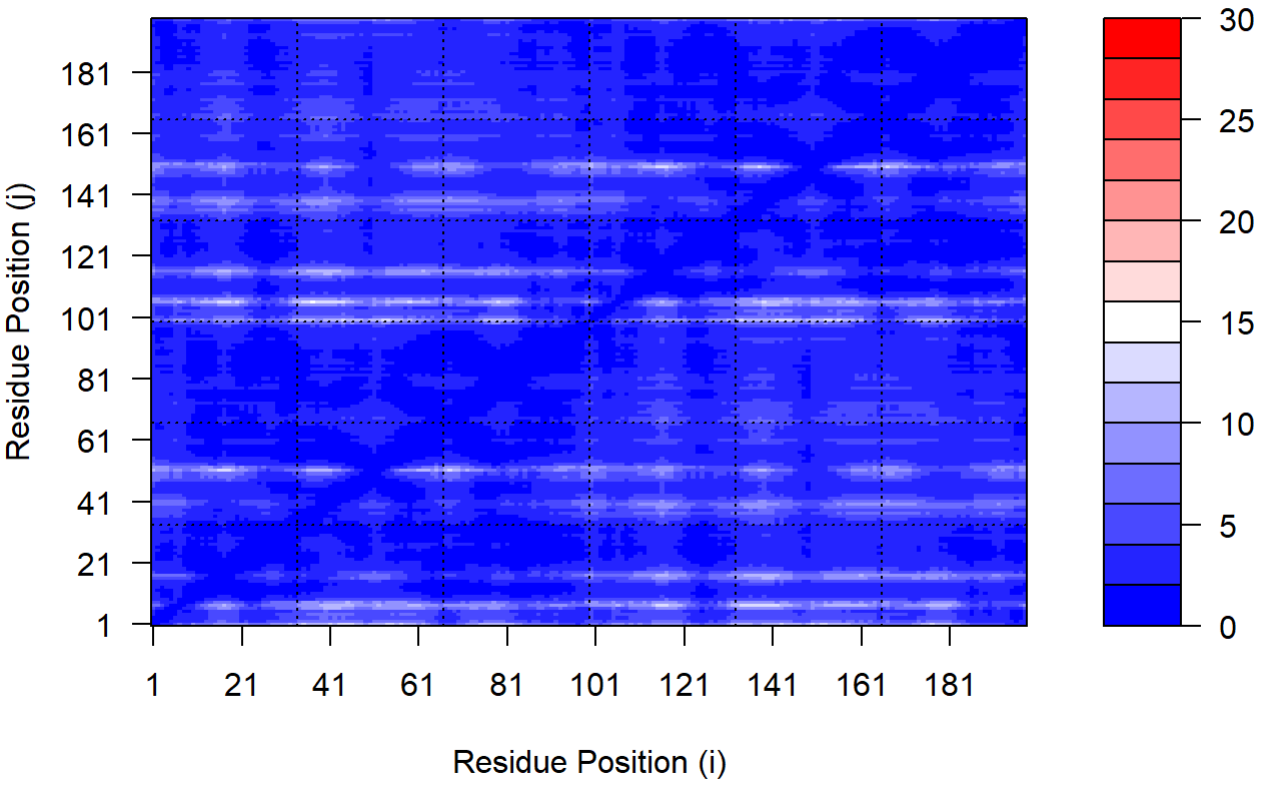
```
plot.dmat(pae1$pae,
          xlab="Residue Position (i)",
          ylab="Residue Position (j)")
```




```
plot.dmat(pae5$pae,
          xlab="Residue Position (i)",
          ylab="Residue Position (j)",
          grid.col = "black",
          zlim=c(0,30))
```




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


Residue Position (j)
181
161
141
121
101
81
61
41
21
1
1
21
41
61
81
101
121
141
161
181
Residue Position (i)
30
25
20
15
10
5
0