

Structural Bioinformatics (pt 2)

Sawyer (PID: A16335277)

AlphaFold has change 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://colab.research.google.com/github/sokrypton/ColabFold/blob/main>

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 strucutres up in Mol*

```
results_dir <- "hivpr1dimer_23119/"

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

# Print our PDB file names
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"

library(bio3d)

# install.packages("BiocManager")
```

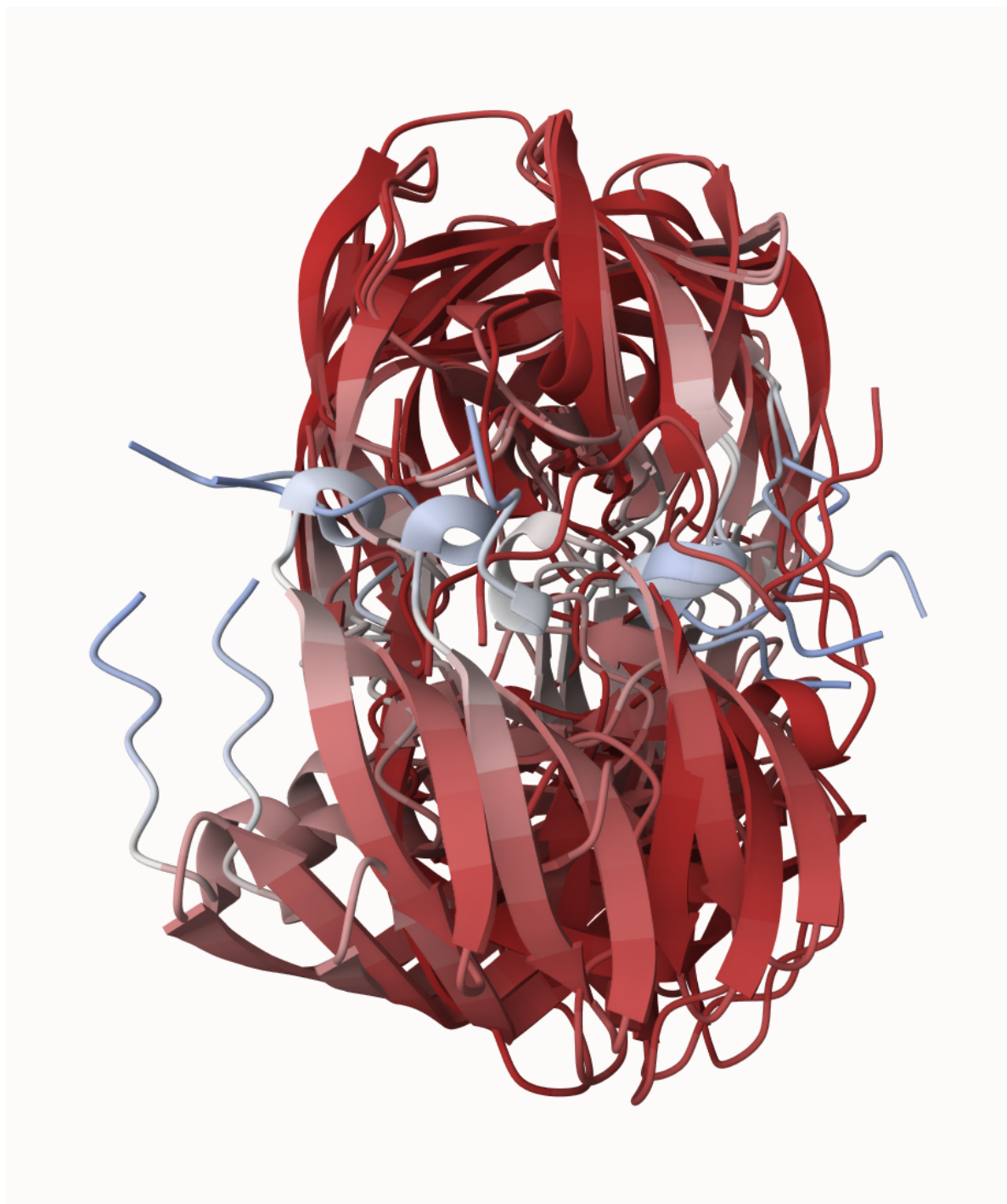


Figure 1: Uncertainty/Disorder superimposed (via mol) *mol* visual

```
# BiocManager::install("msa")

# 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_00
hivpr1dimer_23119/hivpr1dimer_23119_unrelaxed_rank_002_alphafold2_multimer_v3_model_5_seed_00
hivpr1dimer_23119/hivpr1dimer_23119_unrelaxed_rank_003_alphafold2_multimer_v3_model_4_seed_00
hivpr1dimer_23119/hivpr1dimer_23119_unrelaxed_rank_004_alphafold2_multimer_v3_model_2_seed_00
hivpr1dimer_23119/hivpr1dimer_23119_unrelaxed_rank_005_alphafold2_multimer_v3_model_3_seed_00
.....
```

Extracting sequences

```
pdb/seq: 1    name: hivpr1dimer_23119/hivpr1dimer_23119_unrelaxed_rank_001_alphafold2_multimer_v3_model_1_seed_00
pdb/seq: 2    name: hivpr1dimer_23119/hivpr1dimer_23119_unrelaxed_rank_002_alphafold2_multimer_v3_model_5_seed_00
pdb/seq: 3    name: hivpr1dimer_23119/hivpr1dimer_23119_unrelaxed_rank_003_alphafold2_multimer_v3_model_4_seed_00
pdb/seq: 4    name: hivpr1dimer_23119/hivpr1dimer_23119_unrelaxed_rank_004_alphafold2_multimer_v3_model_2_seed_00
pdb/seq: 5    name: hivpr1dimer_23119/hivpr1dimer_23119_unrelaxed_rank_005_alphafold2_multimer_v3_model_3_seed_00
```

pdbs

```
1 . . . . 50
[Truncated_Name:1]hivpr1dime PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:2]hivpr1dime PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:3]hivpr1dime PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:4]hivpr1dime PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:5]hivpr1dime PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
*****
1 . . . . 50

51 . . . . 100
[Truncated_Name:1]hivpr1dime GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:2]hivpr1dime GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:3]hivpr1dime GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:4]hivpr1dime GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:5]hivpr1dime GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
*****
```

```

51          .          .          .          .          100

101         .          .          .          .          150
[Truncated_Name:1]hivpr1dime  QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWPKPMIGGIG
[Truncated_Name:2]hivpr1dime  QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWPKPMIGGIG
[Truncated_Name:3]hivpr1dime  QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWPKPMIGGIG
[Truncated_Name:4]hivpr1dime  QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWPKPMIGGIG
[Truncated_Name:5]hivpr1dime  QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWPKPMIGGIG
*****
101         .          .          .          .          150

151         .          .          .          .          198
[Truncated_Name:1]hivpr1dime  GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:2]hivpr1dime  GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:3]hivpr1dime  GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:4]hivpr1dime  GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:5]hivpr1dime  GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
*****
151         .          .          .          .          198

```

Call:

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

Class:

```
pdb, fasta
```

Alignment dimensions:

```
5 sequence rows; 198 position columns (198 non-gap, 0 gap)
```

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

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

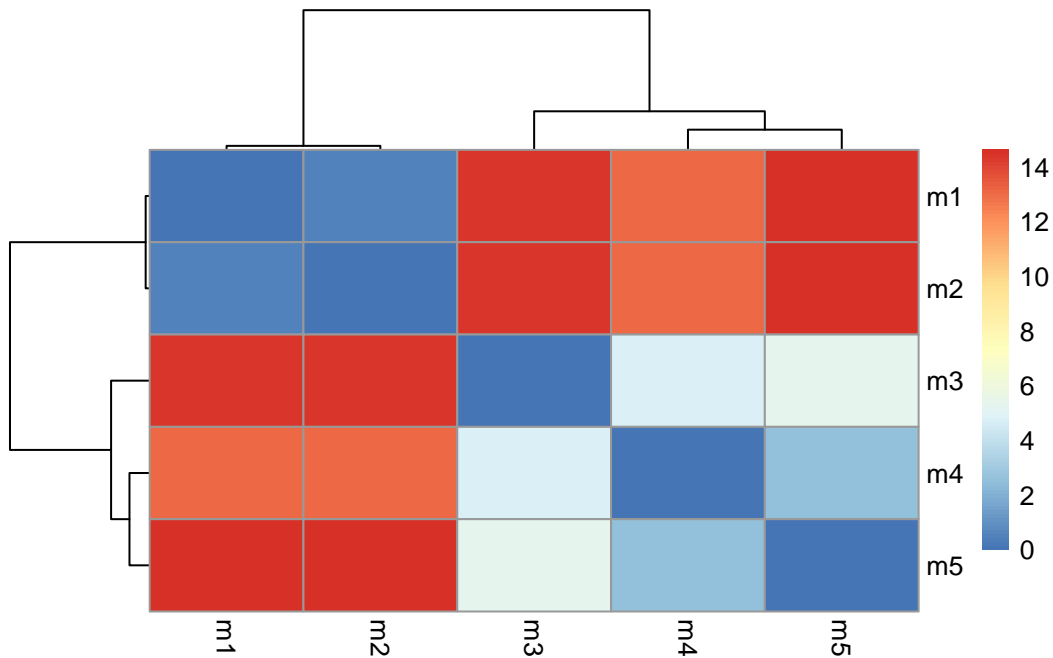
Warning in rmsd(pdb, fit = T): No indices provided, using the 198 non NA positions

```
range(rd)
```

```
[1] 0.000 14.631
```

```
# install.packages('pheatmap')
library(pheatmap)

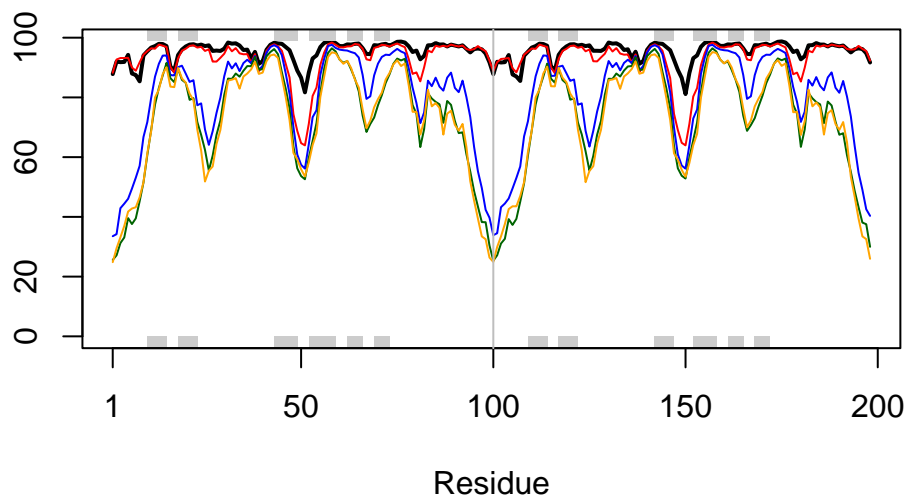
colnames(rd) <- paste0("m",1:5)
rownames(rd) <- paste0("m",1:5)
pheatmap(rd)
```



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

Note: Accessing on-line PDB file

```
plotb3(pdb$b[1,], typ="l", lwd=2, sse=pdb)
points(pdb$b[2,], typ="l", col="red")
points(pdb$b[3,], typ="l", col="blue")
points(pdb$b[4,], typ="l", col="darkgreen")
points(pdb$b[5,], typ="l", col="orange")
abline(v=100, col="gray")
```



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

```
core size 197 of 198 vol = 4578.336
core size 196 of 198 vol = 3931.103
core size 195 of 198 vol = 3709.727
core size 194 of 198 vol = 3496.014
core size 193 of 198 vol = 3302.428
core size 192 of 198 vol = 3146.468
core size 191 of 198 vol = 3048.959
core size 190 of 198 vol = 2970.348
core size 189 of 198 vol = 2893.007
core size 188 of 198 vol = 2831.818
core size 187 of 198 vol = 2774.499
core size 186 of 198 vol = 2728.035
core size 185 of 198 vol = 2704.937
core size 184 of 198 vol = 2701.97
core size 183 of 198 vol = 2715.897
core size 182 of 198 vol = 2809.84
core size 181 of 198 vol = 2888.937
core size 180 of 198 vol = 2967.269
core size 179 of 198 vol = 3036.243
core size 178 of 198 vol = 3066.274
```

core size 177 of 198	vol = 3096.82
core size 176 of 198	vol = 3056.401
core size 175 of 198	vol = 3014.755
core size 174 of 198	vol = 2974.999
core size 173 of 198	vol = 2898.037
core size 172 of 198	vol = 2810.159
core size 171 of 198	vol = 2747.518
core size 170 of 198	vol = 2684.42
core size 169 of 198	vol = 2620.339
core size 168 of 198	vol = 2550.863
core size 167 of 198	vol = 2492.567
core size 166 of 198	vol = 2422.963
core size 165 of 198	vol = 2358.901
core size 164 of 198	vol = 2298.277
core size 163 of 198	vol = 2235.903
core size 162 of 198	vol = 2171.006
core size 161 of 198	vol = 2093.544
core size 160 of 198	vol = 2029.129
core size 159 of 198	vol = 1950.943
core size 158 of 198	vol = 1881.001
core size 157 of 198	vol = 1801.491
core size 156 of 198	vol = 1728.877
core size 155 of 198	vol = 1660.022
core size 154 of 198	vol = 1586.134
core size 153 of 198	vol = 1532.702
core size 152 of 198	vol = 1460.171
core size 151 of 198	vol = 1399.236
core size 150 of 198	vol = 1333.893
core size 149 of 198	vol = 1271.731
core size 148 of 198	vol = 1219.48
core size 147 of 198	vol = 1175.987
core size 146 of 198	vol = 1138.462
core size 145 of 198	vol = 1102.108
core size 144 of 198	vol = 1049.627
core size 143 of 198	vol = 1014.047
core size 142 of 198	vol = 970.56
core size 141 of 198	vol = 929.163
core size 140 of 198	vol = 889.089
core size 139 of 198	vol = 846.653
core size 138 of 198	vol = 805.785
core size 137 of 198	vol = 775.019
core size 136 of 198	vol = 743.075
core size 135 of 198	vol = 715.68

core size 134 of 198 vol = 689.773
core size 133 of 198 vol = 660.314
core size 132 of 198 vol = 630.951
core size 131 of 198 vol = 597.191
core size 130 of 198 vol = 566.973
core size 129 of 198 vol = 532.874
core size 128 of 198 vol = 496.192
core size 127 of 198 vol = 463.167
core size 126 of 198 vol = 431.877
core size 125 of 198 vol = 408.848
core size 124 of 198 vol = 376.594
core size 123 of 198 vol = 362.36
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

```

```
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

```

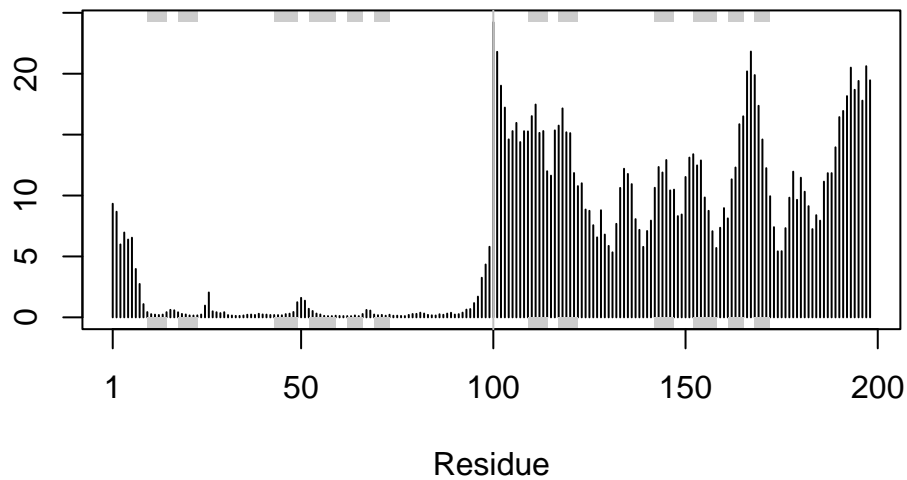
```
xyz <- pdbfit(pdb, core.inds, outpath="corefit_structures")
```

```
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)

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
```

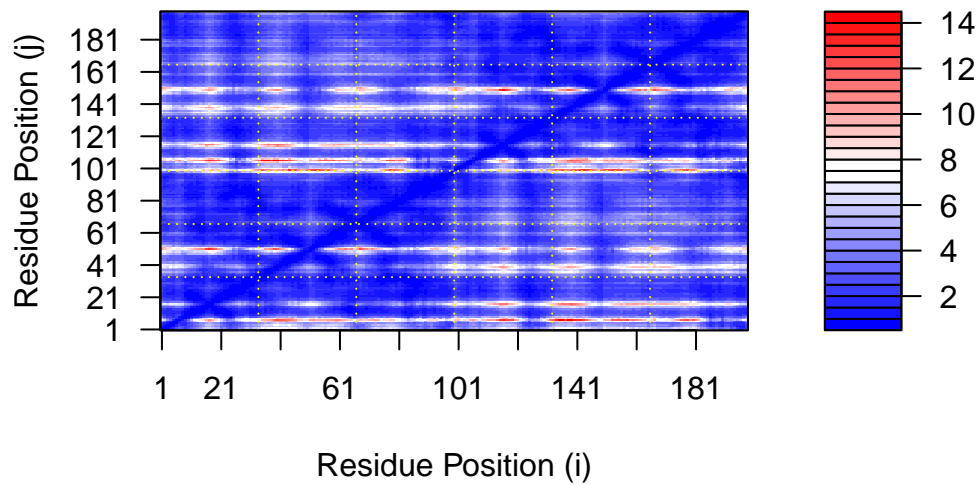
```
pae1$max_pae
```

```
[1] 14.09375
```

```
pae5$max_pae
```

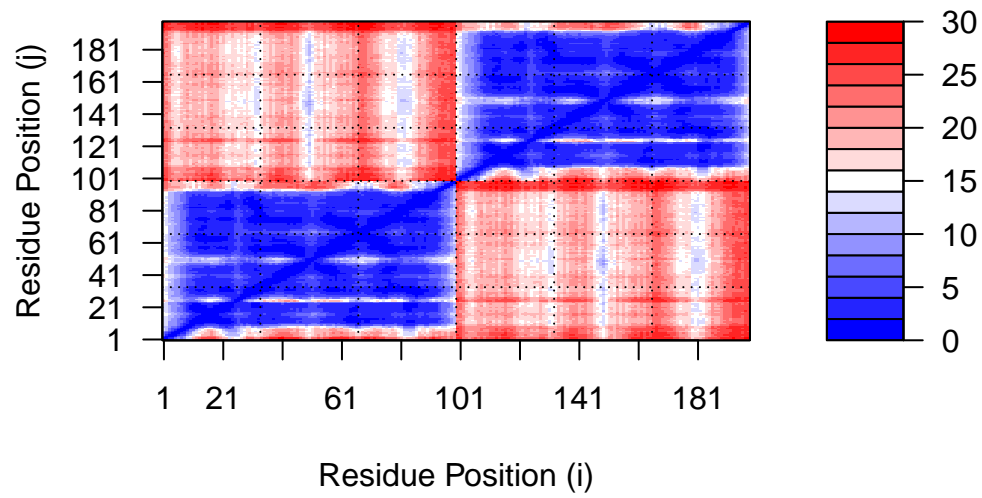
```
[1] 29.29688
```

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

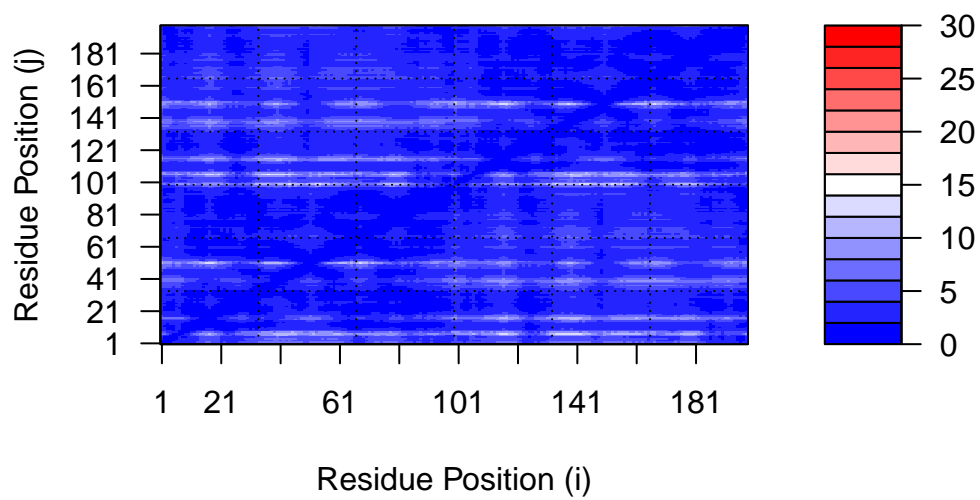


```
plot.dmat(pae5$pae,  
          xlab="Residue Position (i)",  
          ylab="Residue Position (j)",  
          grid.col = "black",
```

```
zlim=c(0,30))
```



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



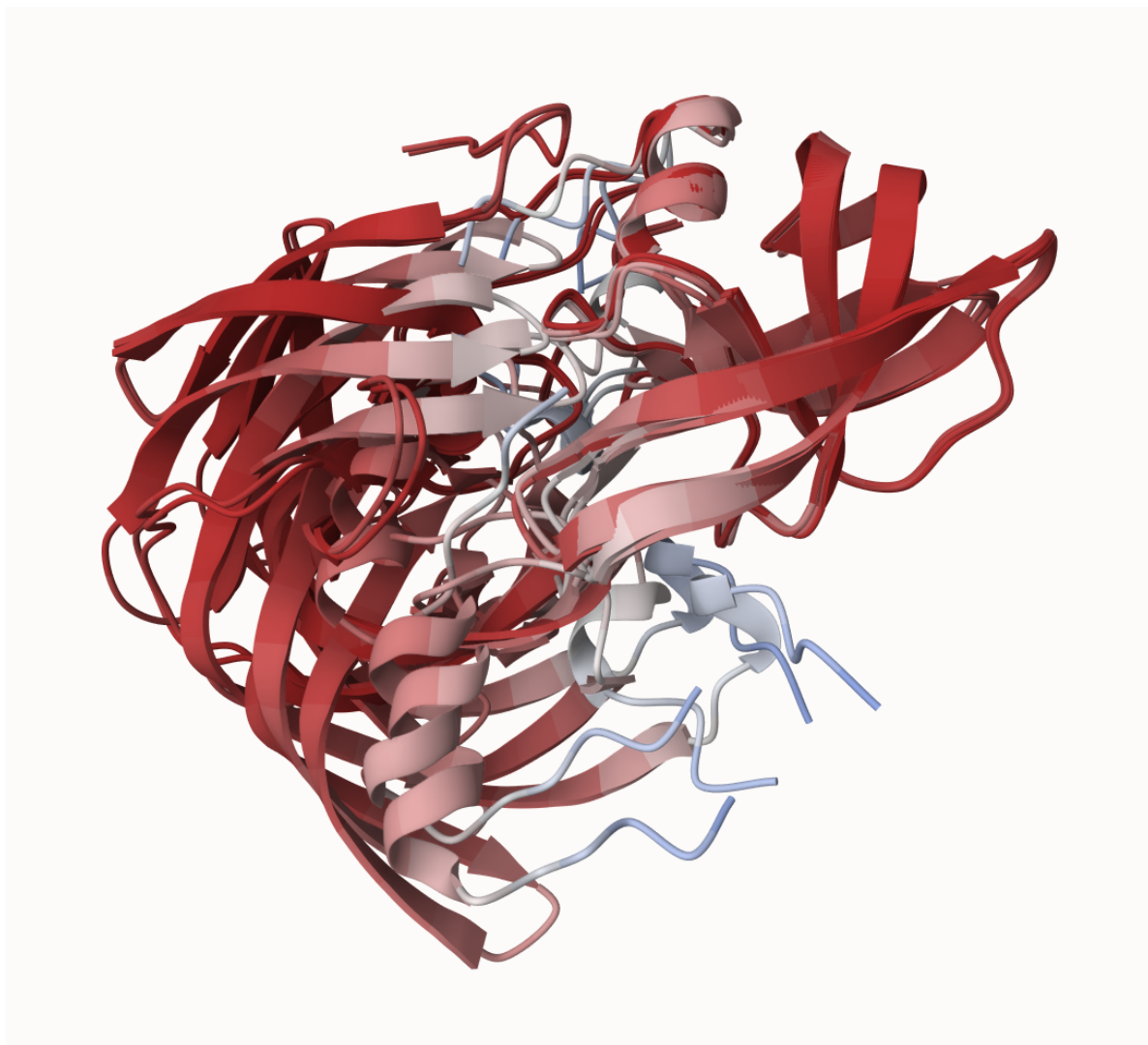


Figure 2: Uncertainty/Disorder superimposed (via R) mol* visual