Class 11 Structural Bioinformatics (pt2 AlphaFold)

AUTHOR
Erin McTavish PID: A17300519

Background

We saw last day that the main repository for biomolecular structure (the PDB database) only has ~250,000 entries.

UnitProtKB (the main protein sequence database) has over 200 million entries!

AlphaFold

In this hands-on session we will utilize AlphaFold to predict protein structure ffrom sequence (Jumper et al. 2021).

Without the aid of such approaches, it can take years of expensive laboratory work to determine the structure of just one protein. With AlphaFold we can now accurately compute a typical protein structure in as little as ten minutes.

The EBI AlpaFold database

The EBI AlphaFold database contains lots of computed structure models. It is increasingly likely that the structure you are interested in is already on this database.< https:://alphafold.ebi.ac.uk >

There are 3 major outputs from AlphaFold

1. A model of structure in **PDB** format.
2. A **pLDDT score**: that tells us how canfident the model is for a given residue in your protein (High values are good, above 70)
3. A **PAE score** that tells us about protein packing quality.

If you can’t find a matching entry for the sequence you ar einterested in AFDB you can run AlphaFold yourself…

Running AlphaFold

We will use ColabFold to run AlphaFold on our sequence <

https:://github.com/sokrytpon/ColabFold >

Figure from AlphaFold here!

Interpreting Results

Custom analysis of resulting models

We can read all the AlphaFold results into R and do more quantitive analysis than just view the structures in Mol-star:

Read all the PDB models:

```
library(bio3d)
p <- read.pdb("hivpr_23119_0/hivpr_23119_0_unrelaxed_rank_001_alphafold2_multimer_v3_mode
p
```

```
 Call:  read.pdb(file =
"hivpr_23119_0/hivpr_23119_0_unrelaxed_rank_001_alphafold2_multimer_v3_model_4_seed_000.p
db")

   Total Models#: 1
     Total Atoms#: 1514,  XYZs#: 4542  Chains#: 2  (values: A B)

     Protein Atoms#: 1514  (residues/Calpha atoms#: 198)
     Nucleic acid Atoms#: 0  (residues/phosphate atoms#: 0)

     Non-protein/nucleic Atoms#: 0  (residues: 0)
     Non-protein/nucleic resid values: [ none ]

   Protein sequence:
      PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIGGFIKVRQYD
      QILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFPQITLWQRPLVTIKIGGQLKE
      ALLDTGADDTVLEEMSLPGRWKPKMIGGIGGFIKVRQYDQILIEICGHKAIGTVLVGPTP
      VNIIGRNLLTQIGCTLNF

+ attr: atom, xyz, calpha, call
```

```r
pdb_files <- list.files("hivpr_23119_0/", pattern = ".pdb", full.names = T)
pdbs <- pdbaln(pdb_files, fit=TRUE, exefile= "msa")
```

```
Reading PDB files:
hivpr_23119_0//hivpr_23119_0_unrelaxed_rank_001_alphafold2_multimer_v3_model_4_seed_000.p
db
hivpr_23119_0//hivpr_23119_0_unrelaxed_rank_002_alphafold2_multimer_v3_model_1_seed_000.p
db
hivpr_23119_0//hivpr_23119_0_unrelaxed_rank_003_alphafold2_multimer_v3_model_5_seed_000.p
db
hivpr_23119_0//hivpr_23119_0_unrelaxed_rank_004_alphafold2_multimer_v3_model_2_seed_000.p
db
hivpr_23119_0//hivpr_23119_0_unrelaxed_rank_005_alphafold2_multimer_v3_model_3_seed_000.p
db
.....

Extracting sequences

pdb/seq: 1   name:
hivpr_23119_0//hivpr_23119_0_unrelaxed_rank_001_alphafold2_multimer_v3_model_4_seed_000.p
db
pdb/seq: 2   name:
hivpr_23119_0//hivpr_23119_0_unrelaxed_rank_002_alphafold2_multimer_v3_model_1_seed_000.p
db
pdb/seq: 3   name:
hivpr_23119_0//hivpr_23119_0_unrelaxed_rank_003_alphafold2_multimer_v3_model_5_seed_000.p
db
pdb/seq: 4   name:
hivpr_23119_0//hivpr_23119_0_unrelaxed_rank_004_alphafold2_multimer_v3_model_2_seed_000.p
db
pdb/seq: 5   name:
```

```
hivpr_23119_0//hivpr_23119_0_unrelaxed_rank_005_alphafold2_multimer_v3_model_3_seed_000.p
db
```

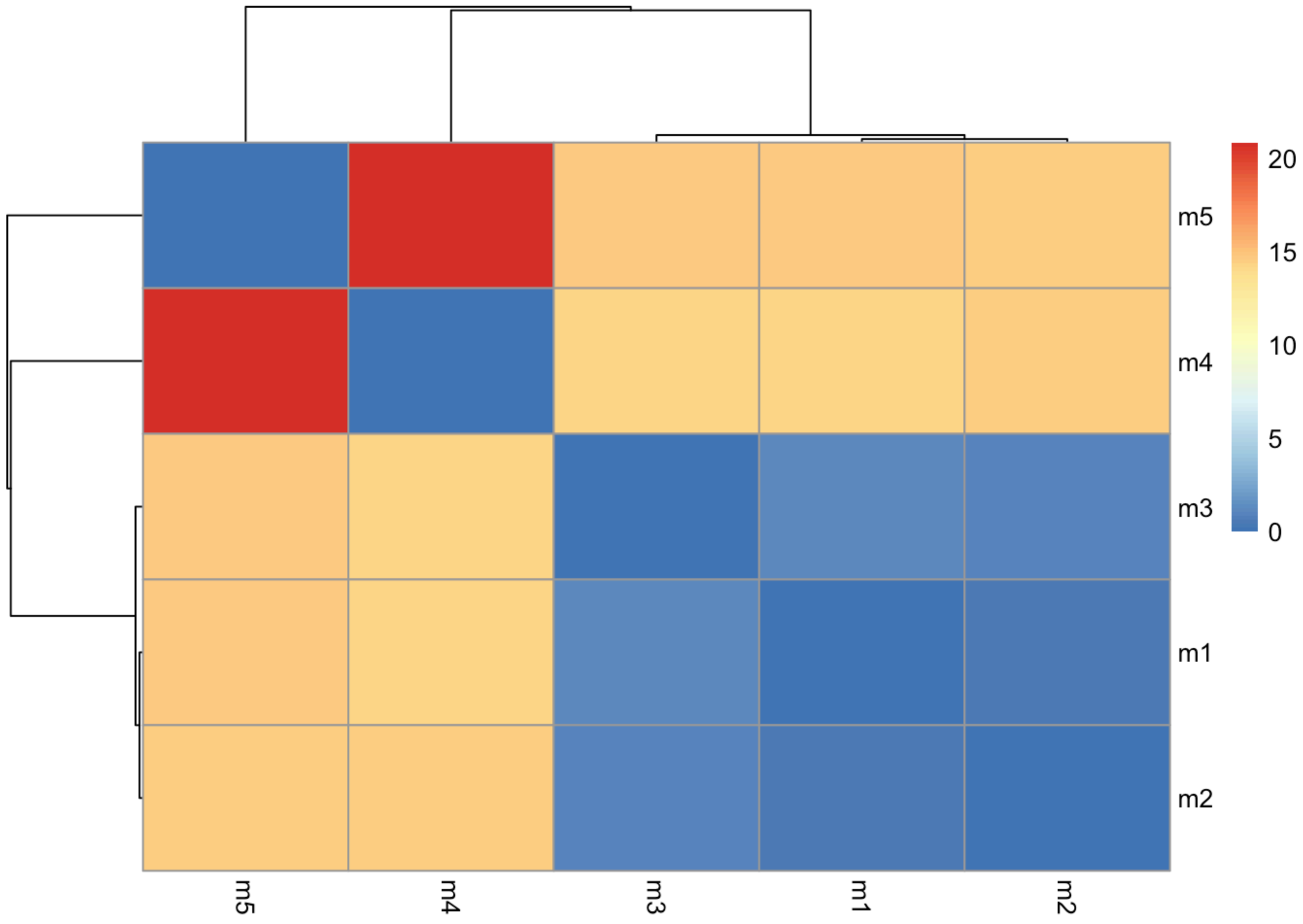
```
#library(bio3dview)
#view.pdbs(pdbs)
```

How similar or different are my models?

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

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

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



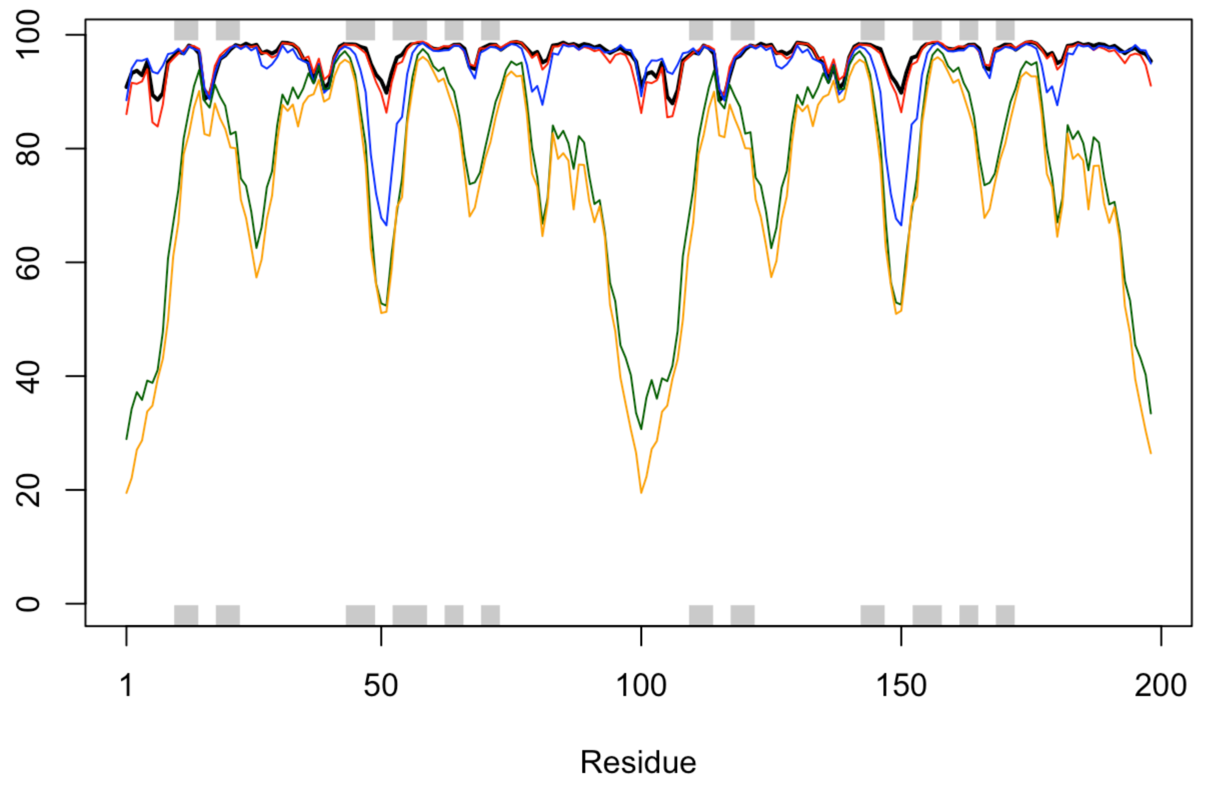
Now lets plot the pLDDT values across all models.

```
library(bio3d)
```

```
# 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")
```



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 = 9885.419
 core size 196 of 198  vol = 6898.241
 core size 195 of 198  vol = 1338.035
 core size 194 of 198  vol = 1040.677
 core size 193 of 198  vol = 951.865
 core size 192 of 198  vol = 899.087
 core size 191 of 198  vol = 834.733
 core size 190 of 198  vol = 771.342
 core size 189 of 198  vol = 733.069
 core size 188 of 198  vol = 697.285
 core size 187 of 198  vol = 659.748
 core size 186 of 198  vol = 625.28
 core size 185 of 198  vol = 589.548
 core size 184 of 198  vol = 568.261
 core size 183 of 198  vol = 545.022
 core size 182 of 198  vol = 512.897
 core size 181 of 198  vol = 490.731
 core size 180 of 198  vol = 470.274
 core size 179 of 198  vol = 450.738
 core size 178 of 198  vol = 434.743
 core size 177 of 198  vol = 420.345
 core size 176 of 198  vol = 406.666
 core size 175 of 198  vol = 393.341
 core size 174 of 198  vol = 382.402
 core size 173 of 198  vol = 372.866
 core size 172 of 198  vol = 357.001
 core size 171 of 198  vol = 346.576
 core size 170 of 198  vol = 337.454
 core size 169 of 198  vol = 326.668
 core size 168 of 198  vol = 314.959
 core size 167 of 198  vol = 304.136
 core size 166 of 198  vol = 294.561
 core size 165 of 198  vol = 285.658
 core size 164 of 198  vol = 278.893
 core size 163 of 198  vol = 266.773
 core size 162 of 198  vol = 259.003
 core size 161 of 198  vol = 247.731
 core size 160 of 198  vol = 239.849
 core size 159 of 198  vol = 234.973
 core size 158 of 198  vol = 230.071
 core size 157 of 198  vol = 221.995
 core size 156 of 198  vol = 215.629
 core size 155 of 198  vol = 206.8
 core size 154 of 198  vol = 196.992
 core size 153 of 198  vol = 188.547
 core size 152 of 198  vol = 182.27
 core size 151 of 198  vol = 176.961
 core size 150 of 198  vol = 170.72
 core size 149 of 198  vol = 166.128
 core size 148 of 198  vol = 159.805
 core size 147 of 198  vol = 153.775
 core size 146 of 198  vol = 149.101
 core size 145 of 198  vol = 143.664
 core size 144 of 198  vol = 137.145
 core size 143 of 198  vol = 132.523
 core size 142 of 198  vol = 127.237
 core size 141 of 198  vol = 121.579
 core size 140 of 198  vol = 116.78
 core size 139 of 198  vol = 112.575 
 core size 138 of 198  vol = 108.175 
 core size 137 of 198  vol = 105.137 
 core size 136 of 198  vol = 101.254 
 core size 135 of 198  vol = 97.379 
 core size 134 of 198  vol = 92.978 
 core size 133 of 198  vol = 88.188 
 core size 132 of 198  vol = 84.032 
 core size 131 of 198  vol = 81.902 
 core size 130 of 198  vol = 78.023 
 core size 129 of 198  vol = 75.276 
 core size 128 of 198  vol = 73.057 
 core size 127 of 198  vol = 70.699 
 core size 126 of 198  vol = 68.976 
 core size 125 of 198  vol = 66.707 
 core size 124 of 198  vol = 64.376 
 core size 123 of 198  vol = 61.145 
 core size 122 of 198  vol = 59.029 
 core size 121 of 198  vol = 56.625 
 core size 120 of 198  vol = 54.369 
 core size 119 of 198  vol = 51.826 
 core size 118 of 198  vol = 49.651 
 core size 117 of 198  vol = 48.19 
 core size 116 of 198  vol = 46.644 
 core size 115 of 198  vol = 44.748 
 core size 114 of 198  vol = 43.288 
 core size 113 of 198  vol = 41.089 
 core size 112 of 198  vol = 39.143 
 core size 111 of 198  vol = 36.468 
 core size 110 of 198  vol = 34.114 
 core size 109 of 198  vol = 31.467 
 core size 108 of 198  vol = 29.445 
 core size 107 of 198  vol = 27.323 
 core size 106 of 198  vol = 25.82 
 core size 105 of 198  vol = 24.149 
 core size 104 of 198  vol = 22.647 
 core size 103 of 198  vol = 21.068 
 core size 102 of 198  vol = 19.953 
 core size 101 of 198  vol = 18.3 
 core size 100 of 198  vol = 15.723 
 core size 99 of 198  vol = 14.841 
 core size 98 of 198  vol = 11.646 
 core size 97 of 198  vol = 9.435 
 core size 96 of 198  vol = 7.354 
 core size 95 of 198  vol = 6.181 
 core size 94 of 198  vol = 5.667 
 core size 93 of 198  vol = 4.706 
 core size 92 of 198  vol = 3.664 
 core size 91 of 198  vol = 2.77 
 core size 90 of 198  vol = 2.151 
 core size 89 of 198  vol = 1.715 
 core size 88 of 198  vol = 1.15
 core size 87 of 198  vol = 0.874
 core size 86 of 198  vol = 0.685
 core size 85 of 198  vol = 0.528
 core size 84 of 198  vol = 0.37
 FINISHED: Min vol ( 0.5 ) reached
```

We can now use the identified core atom positions as a basis for a more suitable superposition and write out the fitted structures to a directory called corefit_structures:

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

```
# 85 positions (cumulative volume <= 0.5 Angstrom^3)
  start end length
1     9  49     41
2    52  95     44
```

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

```
library(bio3d)

# Check current working directory
getwd()
```

```
[1] "/Users/erin/Documents/BIMM 143/class11/class11"
```

```
# Create output directory if it does not already exist
dir.create("corefit_structures", showWarnings = FALSE)

# Confirm directory now exists
list.files()
```

```
 [1] "aln.fa"
 [2] "class11_files"
 [3] "class11.html"
 [4] "class11.qmd"
 [5] "class11.rmarkdown"
 [6] "class11.Rproj"
 [7] "corefit_structures"
 [8] "hivpr_23119_0"
 [9]
"HIVPR_23119_0_HIVPR_23119_0_UNRELAXED_RANK_002_ALPHAFOLD2_MULTIMER_V3_MODEL_1_SEED_000.P
DB-
HIVPR_23119_0_HIVPR_23119_0_UNRELAXED_RANK_003_ALPHAFOLD2_MULTIMER_V3_MODEL_5_SEED_000.PD
B-HIVPR_23119_0_HIVPR_23119_0_UNRELAXED_RANK_001_ALPHAFOLD2_.png"
[10]
"HIVPR_23119_0_HIVPR_23119_0_UNRELAXED_RANK_002_ALPHAFOLD2_MULTIMER_V3_MODEL_1_SEED_000.P
DB-
HIVPR_23119_0_HIVPR_23119_0_UNRELAXED_RANK_003_ALPHAFOLD2_MULTIMER_V3_MODEL_5_SEED_000.PD
```

```
B-HIVPR_23119_0_HIVPR_23119_0_UNRELAXED_RANK_001_ALPHAFOLD2_.png.zip"
[11]
"HIVPR_23119_0_UNRELAXED_RANK_001_ALPHAFOLD2_MULTIMER_V3_MODEL_4_SEED_000.PDB_FLSQ.PDB-
HIVPR_23119_0_UNRELAXED_RANK_002_ALPHAFOLD2_MULTIMER_V3_MODEL_1_SEED_000.PDB_FLSQ.PDB-
HIVPR_23119_0_UNRELAXED_RANK_003_ALPHAFOLD2_MULTIMER_V3_MODEL_5_SEED.png"
[12] "hivpr_23119_0.result.zip"
[13] "m1_conserv.pdb"
[14] "M1_CONSERV.PDB.png"
```

```
# Find the rigid core across all models
core <- core.find(pdbs)
```

```
 core size 197 of 198  vol = 9885.419
 core size 196 of 198  vol = 6898.241
 core size 195 of 198  vol = 1338.035
 core size 194 of 198  vol = 1040.677
 core size 193 of 198  vol = 951.865
 core size 192 of 198  vol = 899.087
 core size 191 of 198  vol = 834.733
 core size 190 of 198  vol = 771.342
 core size 189 of 198  vol = 733.069
 core size 188 of 198  vol = 697.285
 core size 187 of 198  vol = 659.748
 core size 186 of 198  vol = 625.28
 core size 185 of 198  vol = 589.548
 core size 184 of 198  vol = 568.261
 core size 183 of 198  vol = 545.022
 core size 182 of 198  vol = 512.897
 core size 181 of 198  vol = 490.731
 core size 180 of 198  vol = 470.274
 core size 179 of 198  vol = 450.738
 core size 178 of 198  vol = 434.743
 core size 177 of 198  vol = 420.345
 core size 176 of 198  vol = 406.666
 core size 175 of 198  vol = 393.341
 core size 174 of 198  vol = 382.402
 core size 173 of 198  vol = 372.866
 core size 172 of 198  vol = 357.001
 core size 171 of 198  vol = 346.576
 core size 170 of 198  vol = 337.454
 core size 169 of 198  vol = 326.668
 core size 168 of 198  vol = 314.959
 core size 167 of 198  vol = 304.136
 core size 166 of 198  vol = 294.561
 core size 165 of 198  vol = 285.658
 core size 164 of 198  vol = 278.893
 core size 163 of 198  vol = 266.773
 core size 162 of 198  vol = 259.003
 core size 161 of 198  vol = 247.731
 core size 160 of 198  vol = 239.849
```

```
    core size 159 of 198  vol = 234.973
    core size 158 of 198  vol = 230.071
    core size 157 of 198  vol = 221.995
    core size 156 of 198  vol = 215.629
    core size 155 of 198  vol = 206.8
    core size 154 of 198  vol = 196.992
    core size 153 of 198  vol = 188.547
    core size 152 of 198  vol = 182.27
    core size 151 of 198  vol = 176.961
    core size 150 of 198  vol = 170.72
    core size 149 of 198  vol = 166.128
    core size 148 of 198  vol = 159.805
    core size 147 of 198  vol = 153.775
    core size 146 of 198  vol = 149.101
    core size 145 of 198  vol = 143.664
    core size 144 of 198  vol = 137.145
    core size 143 of 198  vol = 132.523
    core size 142 of 198  vol = 127.237
    core size 141 of 198  vol = 121.579
    core size 140 of 198  vol = 116.78
    core size 139 of 198  vol = 112.575
    core size 138 of 198  vol = 108.175
    core size 137 of 198  vol = 105.137
    core size 136 of 198  vol = 101.254
    core size 135 of 198  vol = 97.379
    core size 134 of 198  vol = 92.978
    core size 133 of 198  vol = 88.188
    core size 132 of 198  vol = 84.032
    core size 131 of 198  vol = 81.902
    core size 130 of 198  vol = 78.023
    core size 129 of 198  vol = 75.276
    core size 128 of 198  vol = 73.057
    core size 127 of 198  vol = 70.699
    core size 126 of 198  vol = 68.976
    core size 125 of 198  vol = 66.707
    core size 124 of 198  vol = 64.376
    core size 123 of 198  vol = 61.145
    core size 122 of 198  vol = 59.029
    core size 121 of 198  vol = 56.625
    core size 120 of 198  vol = 54.369
    core size 119 of 198  vol = 51.826
    core size 118 of 198  vol = 49.651
    core size 117 of 198  vol = 48.19
    core size 116 of 198  vol = 46.644
    core size 115 of 198  vol = 44.748
    core size 114 of 198  vol = 43.288
    core size 113 of 198  vol = 41.089
    core size 112 of 198  vol = 39.143
    core size 111 of 198  vol = 36.468
    core size 110 of 198  vol = 34.114
    core size 109 of 198  vol = 31.467
```

```
 core size 108 of 198  vol = 29.445 
 core size 107 of 198  vol = 27.323 
 core size 106 of 198  vol = 25.82 
 core size 105 of 198  vol = 24.149 
 core size 104 of 198  vol = 22.647 
 core size 103 of 198  vol = 21.068 
 core size 102 of 198  vol = 19.953 
 core size 101 of 198  vol = 18.3 
 core size 100 of 198  vol = 15.723 
 core size 99 of 198  vol = 14.841 
 core size 98 of 198  vol = 11.646 
 core size 97 of 198  vol = 9.435 
 core size 96 of 198  vol = 7.354 
 core size 95 of 198  vol = 6.181 
 core size 94 of 198  vol = 5.667 
 core size 93 of 198  vol = 4.706 
 core size 92 of 198  vol = 3.664 
 core size 91 of 198  vol = 2.77 
 core size 90 of 198  vol = 2.151 
 core size 89 of 198  vol = 1.715 
 core size 88 of 198  vol = 1.15 
 core size 87 of 198  vol = 0.874 
 core size 86 of 198  vol = 0.685 
 core size 85 of 198  vol = 0.528 
 core size 84 of 198  vol = 0.37 
 FINISHED: Min vol ( 0.5 ) reached
```

```r
# Get core atom indices (prints summary to console)
core.inds <- print(core, vol = 0.5)
```

```
# 85 positions (cumulative volume <= 0.5 Angstrom^3) 
  start end length
1     9  49     41
2    52  95     44
```

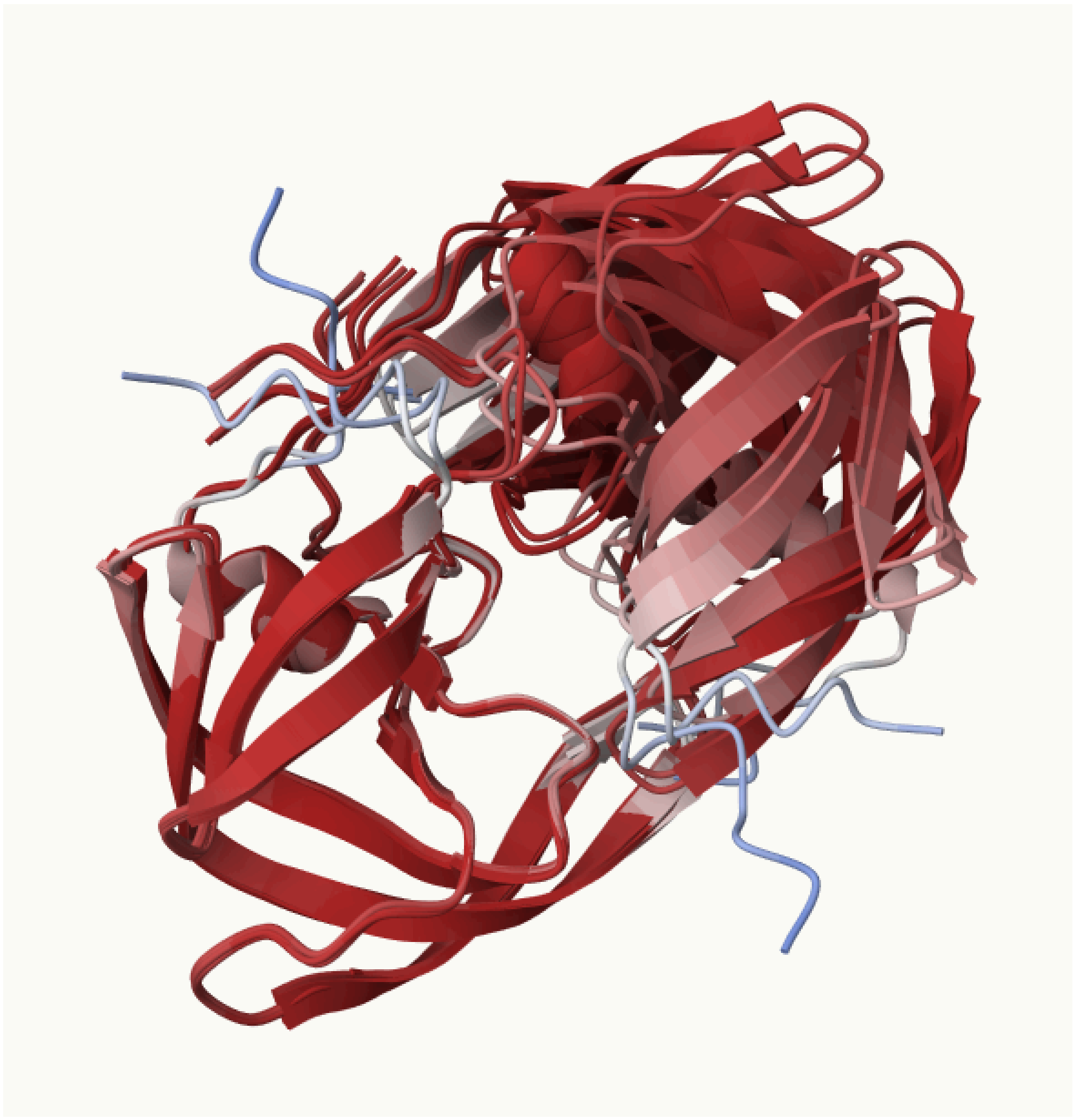
```r
# Fit structures using the core and write PDB files to disk
xyz <- pdbfit(pdbs, core.inds, outpath = "corefit_structures")

# Confirm PDB files were written
list.files("corefit_structures")
```

```
[1]
"hivpr_23119_0_unrelaxed_rank_001_alphafold2_multimer_v3_model_4_seed_000.pdb_flsq.pdb"
[2]
"hivpr_23119_0_unrelaxed_rank_002_alphafold2_multimer_v3_model_1_seed_000.pdb_flsq.pdb"
[3]
"hivpr_23119_0_unrelaxed_rank_003_alphafold2_multimer_v3_model_5_seed_000.pdb_flsq.pdb"
[4]
"hivpr_23119_0_unrelaxed_rank_004_alphafold2_multimer_v3_model_2_seed_000.pdb_flsq.pdb"
```

```
[5]
"hivpr_23119_0_unrelaxed_rank_005_alphafold2_multimer_v3_model_3_seed_000.pdb_flsq.pdb"
```

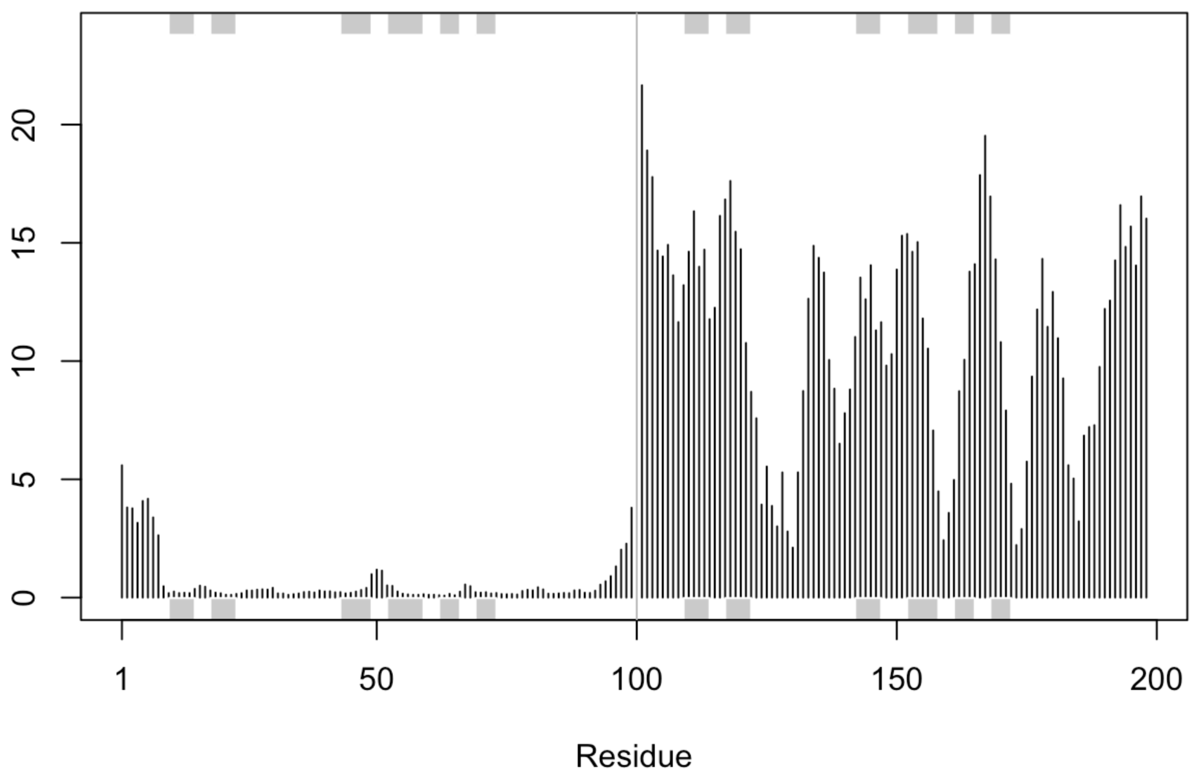
```
system("open corefit_structures")
```



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")
```



The first chain is largely very similar across the different models. However, the second chain is much more variable

##Predicted Alignment Error for domains

Independent of the 3D structure, AlphaFold produces an output called Predicted Aligned Error (PAE). This is detailed in the JSON format result files, one for each model structure.

Below we read these files and see that AlphaFold produces a useful inter-domain prediction for model 1 (and 2) but not for model 5 (or indeed models 3, 4, and 5):

```
library(jsonlite)
results_dir <- "hivpr_23119_0"

pae_files <- list.files(
  path = results_dir,
  pattern = ".*model.*\\.json",
  full.names = TRUE
)
pae_files
```

```
[1]
"hivpr_23119_0/hivpr_23119_0_scores_rank_001_alphafold2_multimer_v3_model_4_seed_000.json
"
```

```
[2]
"hivpr_23119_0/hivpr_23119_0_scores_rank_002_alphafold2_multimer_v3_model_1_seed_000.json
"
[3]
"hivpr_23119_0/hivpr_23119_0_scores_rank_003_alphafold2_multimer_v3_model_5_seed_000.json
"
[4]
"hivpr_23119_0/hivpr_23119_0_scores_rank_004_alphafold2_multimer_v3_model_2_seed_000.json
"
[5]
"hivpr_23119_0/hivpr_23119_0_scores_rank_005_alphafold2_multimer_v3_model_3_seed_000.json
"
```

```
pae1 <- read_json(pae_files[1],simplifyVector = TRUE)
pae2 <- read_json(pae_files[2], simplifyVector = TRUE)
pae3 <- read_json(pae_files[3], simplifyVector = TRUE)
pae4 <- read_json(pae_files[4], 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] 90.81 93.25 93.69 92.88 95.25 89.44
```

The lower the PAE score the better.

```
pae1$max_pae
```

```
[1] 12.84375
```

```
pae2$max_pae
```

```
[1] 20.07812
```

```
pae3$max_pae
```

```
[1] 16.09375
```

```
pae4$max_pae
```

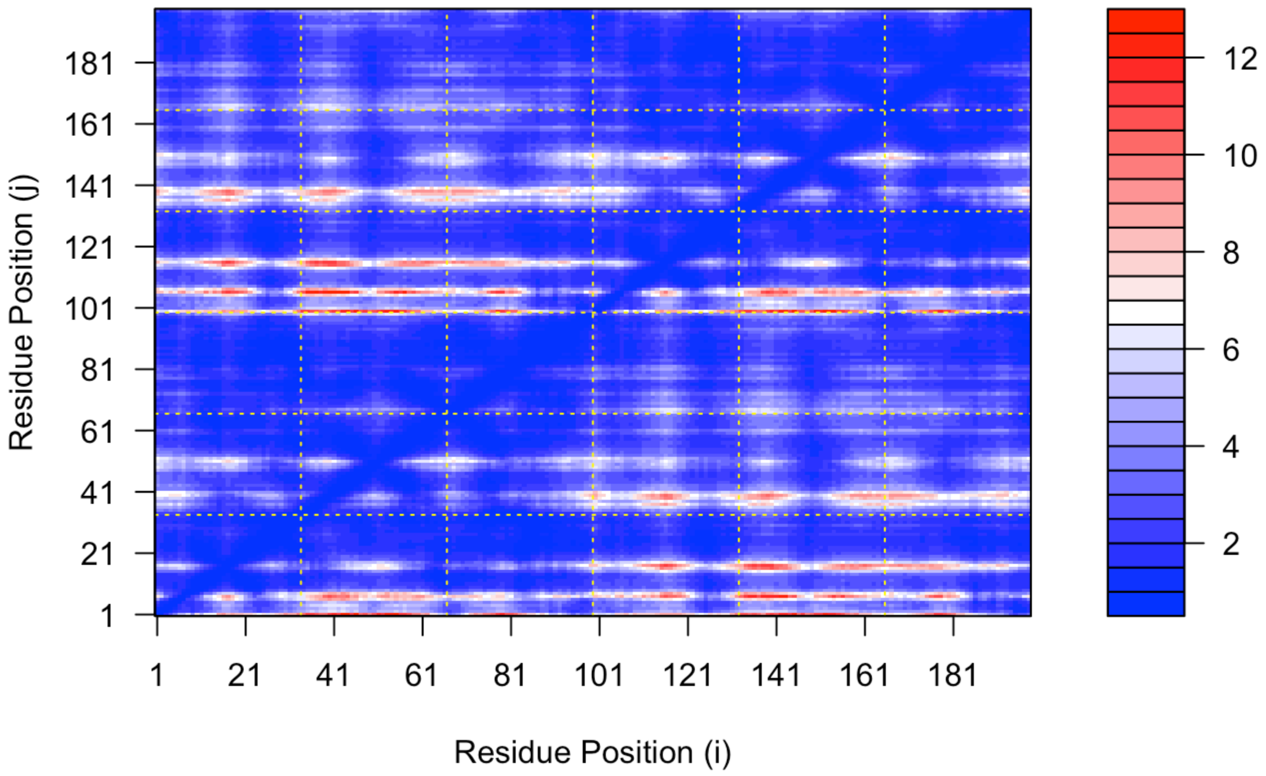
```
[1] 29.60938
```

```
pae5$max_pae
```

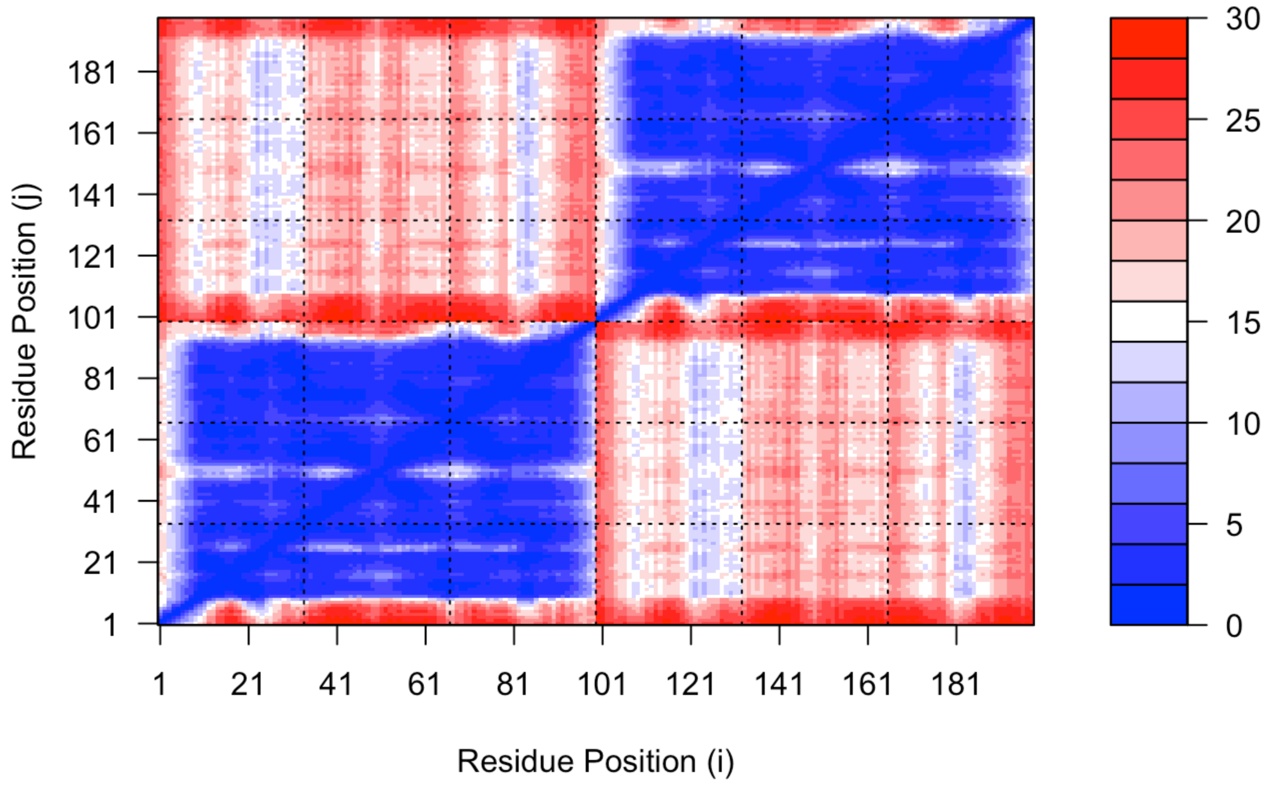
```
[1] 29.59375
```

We can plot the N by N (where N is the number of residues) PAE scores with ggplot or with functions from the Bio3D package:

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

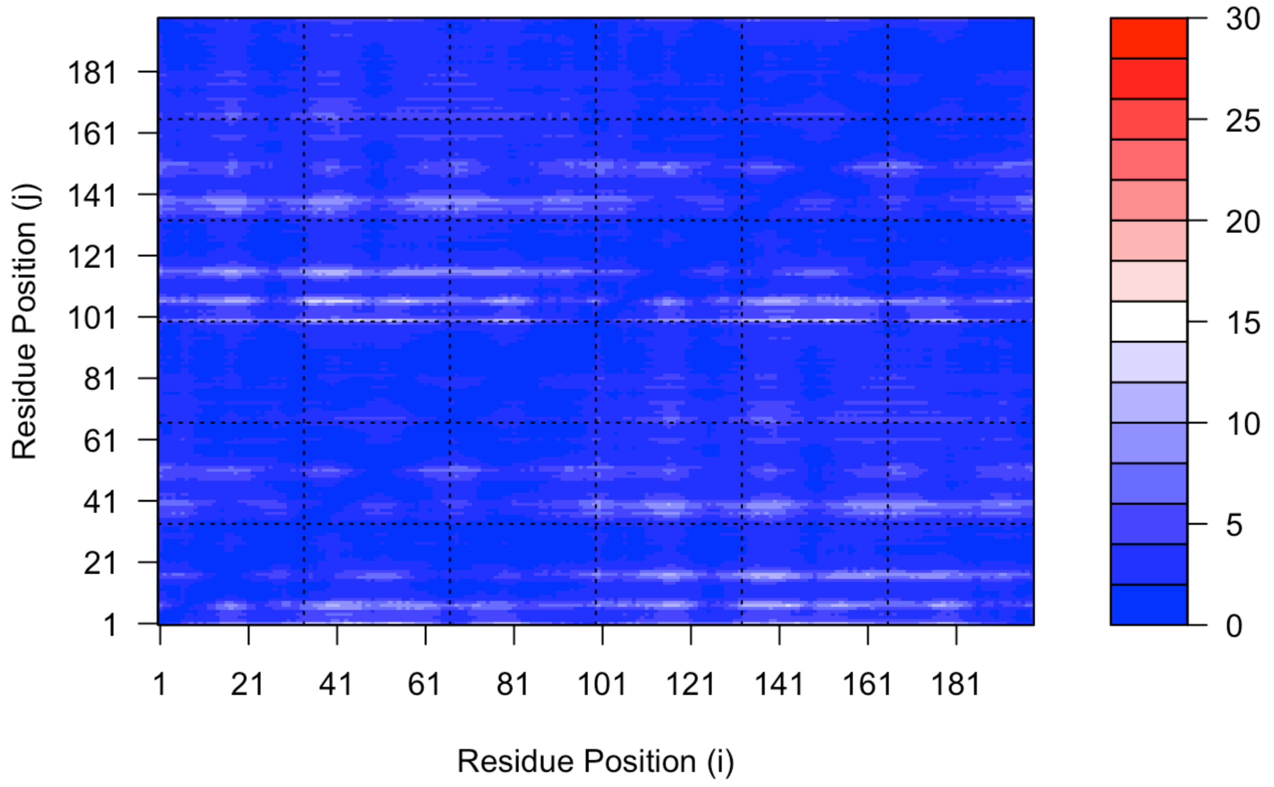


```
plot.dmat(pae5$pae,
          xlab="Residue Position (i)",
          ylab="Residue Position (j)",
          grid.col = "black",
          zlim=c(0,30))
```



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:

```r
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_23119_0/hivpr_23119_0.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

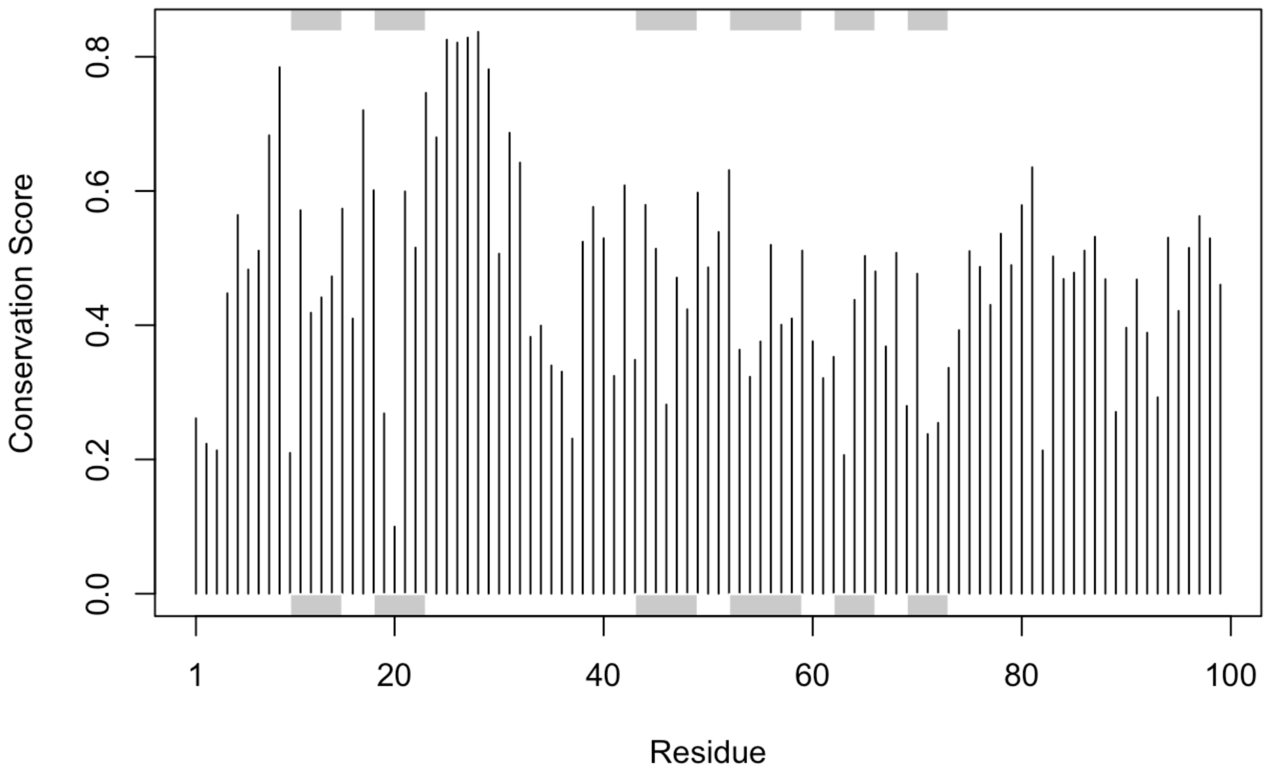
```
dim(aln$ali)
```

```
[1] 5397  132
```

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

```r
sim <- conserv(aln)
```

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



Note the conserved Active Site residues D25, T26, G27, A28. These positions will stand out if we generate a consensus sequence with a high cutoff value:

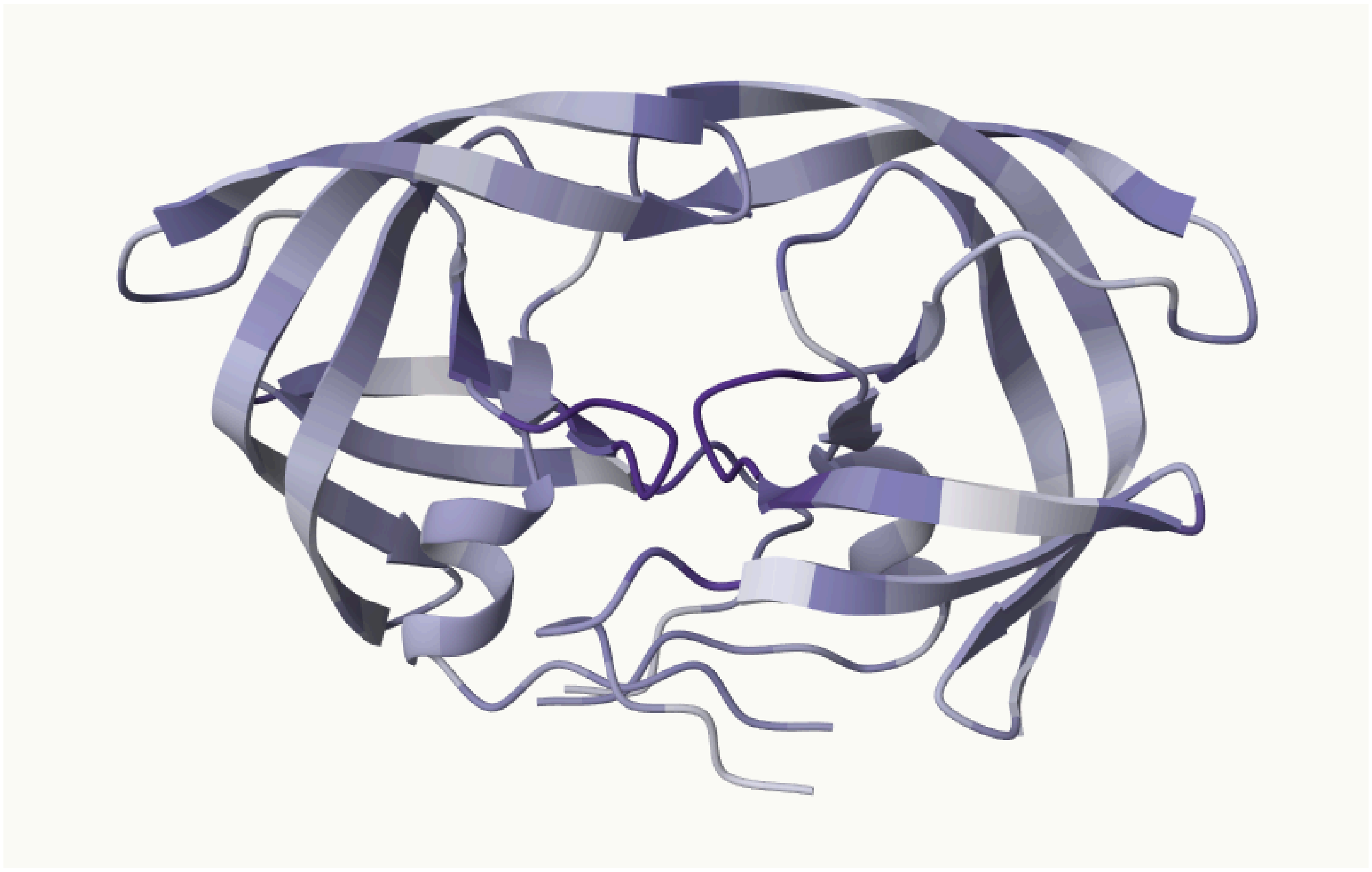
```r
con <- consensus(aln, cutoff = 0.9)
con$seq
```

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

We can map this conservation score to the Occupancy column of a PDB file for viewing in molecular viewer programs

```
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")
```

Here is an image of this data generated from and Mol* using coloring by Occupancy.



We can now clearly see the central conserved active site in this model where the natural peptide substrate (and small molecule inhibitors) would bind between domains