

# BIMM 143 Class 11

Xaler Lu (A17388454)

## Background

The PDB database only has ~250k entries unlike UniProtKB which has over 200 million entries.

We will utilize **AlphaFold** to predict the protein structure from the sequence itself. Atomic structures or distinct 3D shapes of proteins can give insight of its catalysis, signaling, transport, replication, division, and movement. Slight alterations in structure can lead to drastic changes in activity.

We have two approaches: physics-based and knowledge-based.

**Physics-based:** The sum of all potential energy plotted as a PE surface. Energy of bonds = bond stretch, bond angle, and bond rotation. Energy of nonbonds = Electrostatic and van der Waals.

$V(R) = E(\text{bonded}) + E(\text{nonbonded})$  **Knowledge-based:** Docking potentials and use probability as a proxy for energy.

## The EBI AlphaFold Database

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

Use this sequence to search >HIV-Pr PQITLWQRPLVTIKIGGQLKEALLDTGAD-DTVLEEMSLPGRWKPKMIGGIGGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF

There are three major outputs from AlphaFold:

- 1) A **model** of structure in PDB format
- 2) A **pLDDT** score is how confident the model is for a given residue in the protein.
- 3) A **PAE** score is the quality of protein packing.

If you can't find a matching entry for the sequence you are interested in AFDB, then you can run AlphaFold yourself.

## Running AlphaFold

We will run AlphaFold using MMseqs2 on Github. < [https://colab.research.google.com/github/sokrypton/Colab](https://colab.research.google.com/github/sokrypton/ColabFold)  
> Don't forget to use the cloud's GPU because this Mac doesn't have a GPU.



## Alignment



Loading packages

```
library(bio3d)
```

Warning: package 'bio3d' was built under R version 4.3.3

```
library(bio3dview)
library(pheatmap)
```

Warning: package 'pheatmap' was built under R version 4.3.3

## Interpreting Results

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

Read all the PDB models:

```
pdb.files <- list.files("HIVPR_23119/", pattern = ".pdb", full.names = T)

pdbs <- pdbaln(pdb.files, fit = T, exefile = "msa")
```

Reading PDB files:

```
HIVPR_23119//HIVPR_23119_unrelaxed_rank_001_alphafold2_multimer_v3_model_4_seed_000.pdb
HIVPR_23119//HIVPR_23119_unrelaxed_rank_002_alphafold2_multimer_v3_model_1_seed_000.pdb
HIVPR_23119//HIVPR_23119_unrelaxed_rank_003_alphafold2_multimer_v3_model_5_seed_000.pdb
HIVPR_23119//HIVPR_23119_unrelaxed_rank_004_alphafold2_multimer_v3_model_2_seed_000.pdb
HIVPR_23119//HIVPR_23119_unrelaxed_rank_005_alphafold2_multimer_v3_model_3_seed_000.pdb
.....
```

Extracting sequences

```
pdb/seq: 1    name: HIVPR_23119//HIVPR_23119_unrelaxed_rank_001_alphafold2_multimer_v3_model_4
pdb/seq: 2    name: HIVPR_23119//HIVPR_23119_unrelaxed_rank_002_alphafold2_multimer_v3_model_1
pdb/seq: 3    name: HIVPR_23119//HIVPR_23119_unrelaxed_rank_003_alphafold2_multimer_v3_model_5
pdb/seq: 4    name: HIVPR_23119//HIVPR_23119_unrelaxed_rank_004_alphafold2_multimer_v3_model_2
pdb/seq: 5    name: HIVPR_23119//HIVPR_23119_unrelaxed_rank_005_alphafold2_multimer_v3_model_3
```

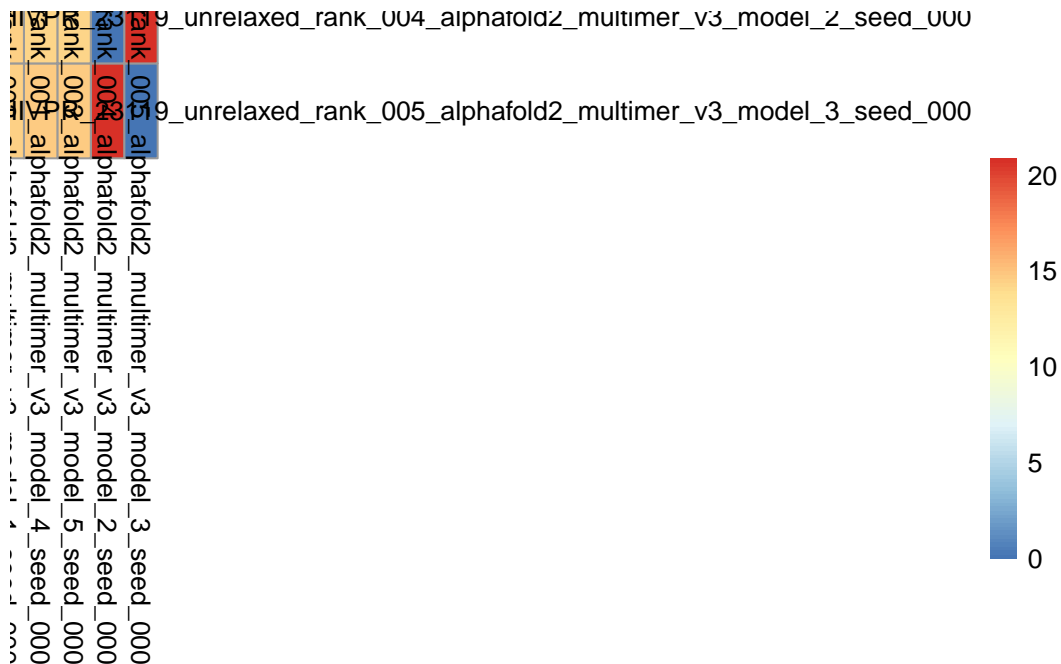
```
#view.pdbs(pdbs)
```

How similar or different are my models? Root mean squared.

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

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

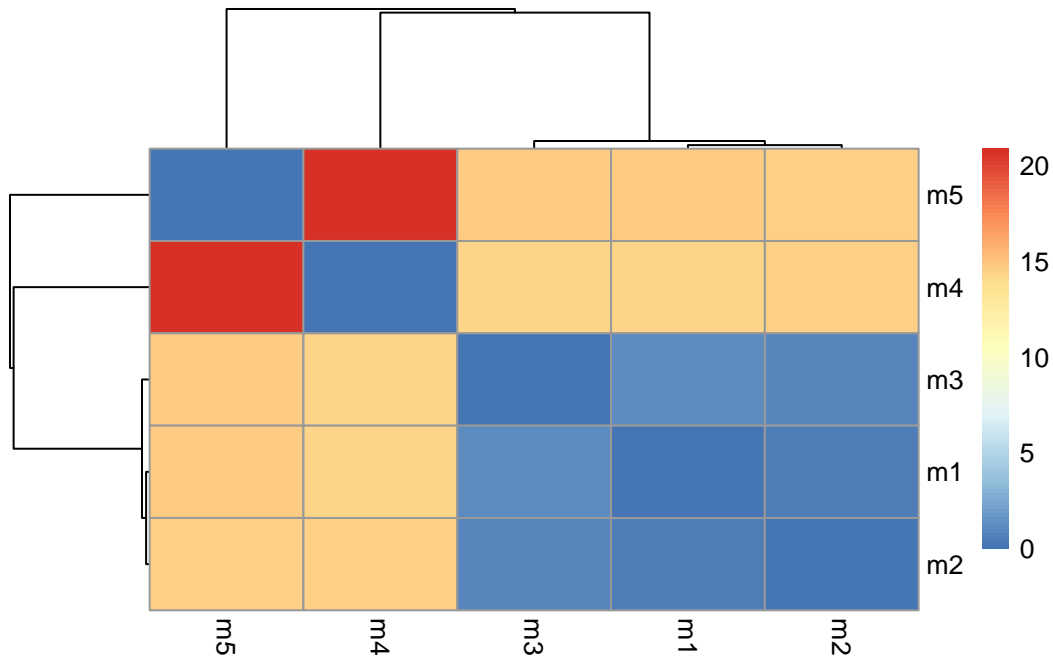
```
pheatmap(rd)
```



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

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

```
pheatmap(rd)
```

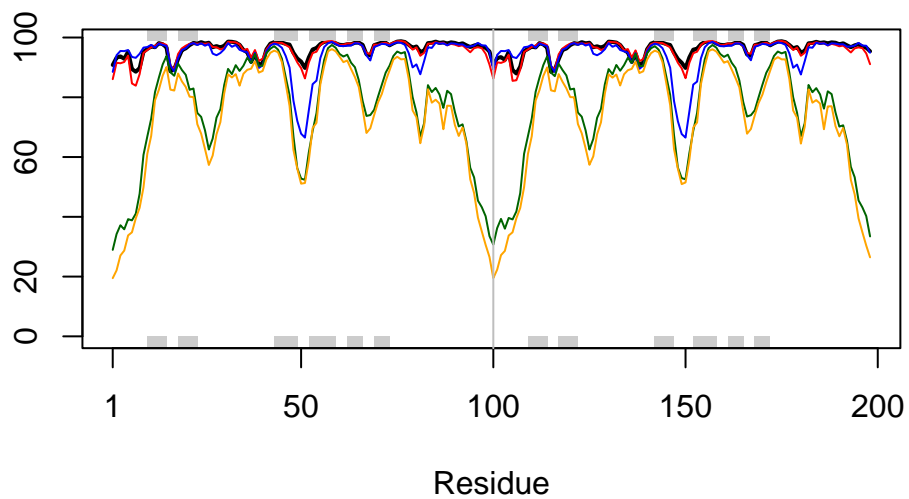


Homework: Make graphs of pLDDT and PAE score.

```
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 = 9885.822
core size 196 of 198  vol = 6896.71
core size 195 of 198  vol = 1337.847
core size 194 of 198  vol = 1040.67
core size 193 of 198  vol = 951.857
core size 192 of 198  vol = 899.083
core size 191 of 198  vol = 834.732
core size 190 of 198  vol = 771.338
core size 189 of 198  vol = 733.065
core size 188 of 198  vol = 697.28
core size 187 of 198  vol = 659.742
core size 186 of 198  vol = 625.273
core size 185 of 198  vol = 589.541
core size 184 of 198  vol = 568.253
core size 183 of 198  vol = 545.015
core size 182 of 198  vol = 512.889
core size 181 of 198  vol = 490.723
core size 180 of 198  vol = 470.266
core size 179 of 198  vol = 450.731
core size 178 of 198  vol = 434.735
```



core size 177 of 198	vol = 420.337
core size 176 of 198	vol = 406.658
core size 175 of 198	vol = 393.334
core size 174 of 198	vol = 382.395
core size 173 of 198	vol = 372.858
core size 172 of 198	vol = 356.994
core size 171 of 198	vol = 346.567
core size 170 of 198	vol = 337.446
core size 169 of 198	vol = 326.659
core size 168 of 198	vol = 314.95
core size 167 of 198	vol = 304.127
core size 166 of 198	vol = 294.552
core size 165 of 198	vol = 285.648
core size 164 of 198	vol = 278.884
core size 163 of 198	vol = 266.765
core size 162 of 198	vol = 258.994
core size 161 of 198	vol = 247.723
core size 160 of 198	vol = 239.84
core size 159 of 198	vol = 234.963
core size 158 of 198	vol = 230.062
core size 157 of 198	vol = 221.985
core size 156 of 198	vol = 215.62
core size 155 of 198	vol = 206.793
core size 154 of 198	vol = 196.984
core size 153 of 198	vol = 188.539
core size 152 of 198	vol = 182.262
core size 151 of 198	vol = 176.954
core size 150 of 198	vol = 170.712
core size 149 of 198	vol = 166.119
core size 148 of 198	vol = 159.796
core size 147 of 198	vol = 153.767
core size 146 of 198	vol = 149.092
core size 145 of 198	vol = 143.657
core size 144 of 198	vol = 137.138
core size 143 of 198	vol = 132.517
core size 142 of 198	vol = 127.231
core size 141 of 198	vol = 121.574
core size 140 of 198	vol = 116.775
core size 139 of 198	vol = 112.57
core size 138 of 198	vol = 108.17
core size 137 of 198	vol = 105.133
core size 136 of 198	vol = 101.249
core size 135 of 198	vol = 97.374

core size 134 of 198	vol = 92.974
core size 133 of 198	vol = 88.184
core size 132 of 198	vol = 84.029
core size 131 of 198	vol = 81.898
core size 130 of 198	vol = 78.019
core size 129 of 198	vol = 75.272
core size 128 of 198	vol = 73.052
core size 127 of 198	vol = 70.695
core size 126 of 198	vol = 68.975
core size 125 of 198	vol = 66.694
core size 124 of 198	vol = 64.394
core size 123 of 198	vol = 62.092
core size 122 of 198	vol = 59.045
core size 121 of 198	vol = 56.629
core size 120 of 198	vol = 54.016
core size 119 of 198	vol = 51.806
core size 118 of 198	vol = 49.652
core size 117 of 198	vol = 48.193
core size 116 of 198	vol = 46.648
core size 115 of 198	vol = 44.752
core size 114 of 198	vol = 43.292
core size 113 of 198	vol = 41.093
core size 112 of 198	vol = 39.147
core size 111 of 198	vol = 36.472
core size 110 of 198	vol = 34.117
core size 109 of 198	vol = 31.47
core size 108 of 198	vol = 29.448
core size 107 of 198	vol = 27.325
core size 106 of 198	vol = 25.822
core size 105 of 198	vol = 24.15
core size 104 of 198	vol = 22.648
core size 103 of 198	vol = 21.069
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 = 9.434
core size 96 of 198	vol = 7.354
core size 95 of 198	vol = 6.179
core size 94 of 198	vol = 5.666
core size 93 of 198	vol = 4.705
core size 92 of 198	vol = 3.665

```

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

```

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

```

```

Total Frames#: 5
Total XYZs#:   594, (Atoms#: 198)

```

```
[1] 16.891 -2.467 -6.562 <...> -14.628 -0.339 -2.293 [2970]
```

```
+ attr: Matrix DIM = 5 x 594
```

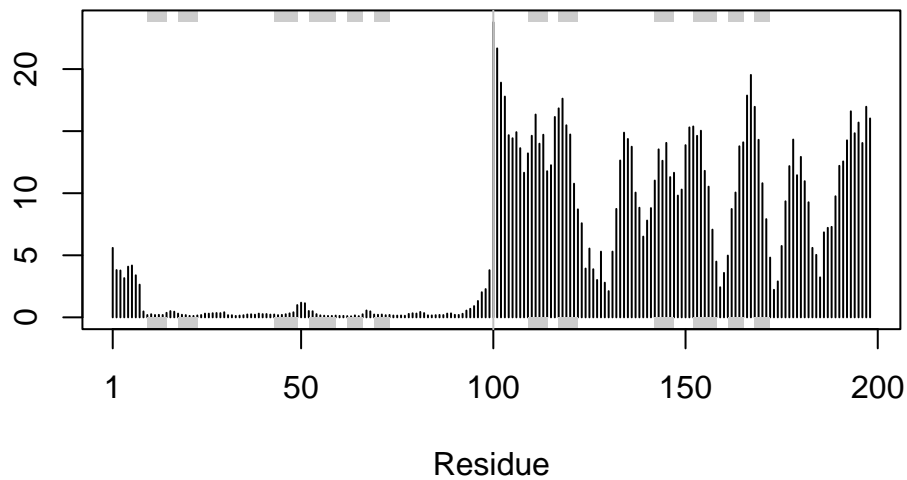
The first chain is more similar across different models, but the second chain is more variable.

```

rf <- rmsf(xyz)

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

```



```
library(jsonlite)
```

Warning: package 'jsonlite' was built under R version 4.3.3

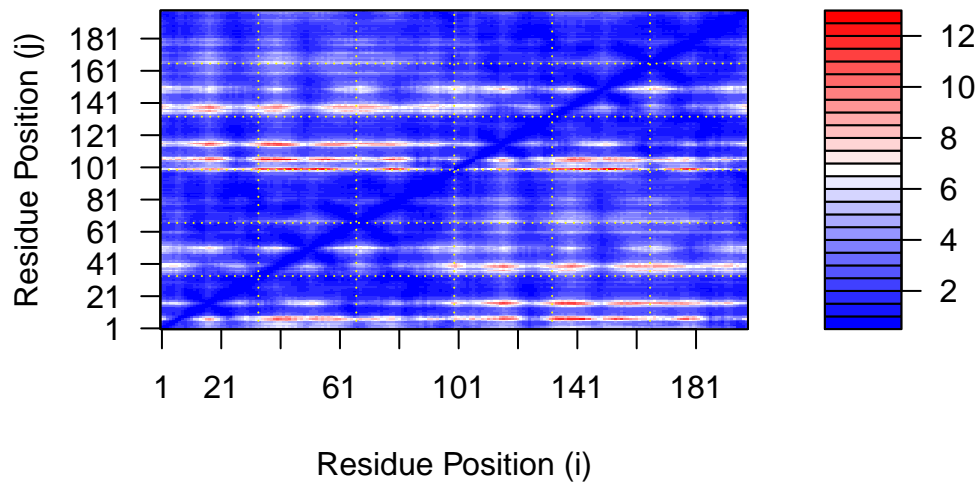
```
pae_files <- list.files(path="HIVPR_23119/",
                        pattern=".*model.*\\.json",
                        full.names = TRUE)
```

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

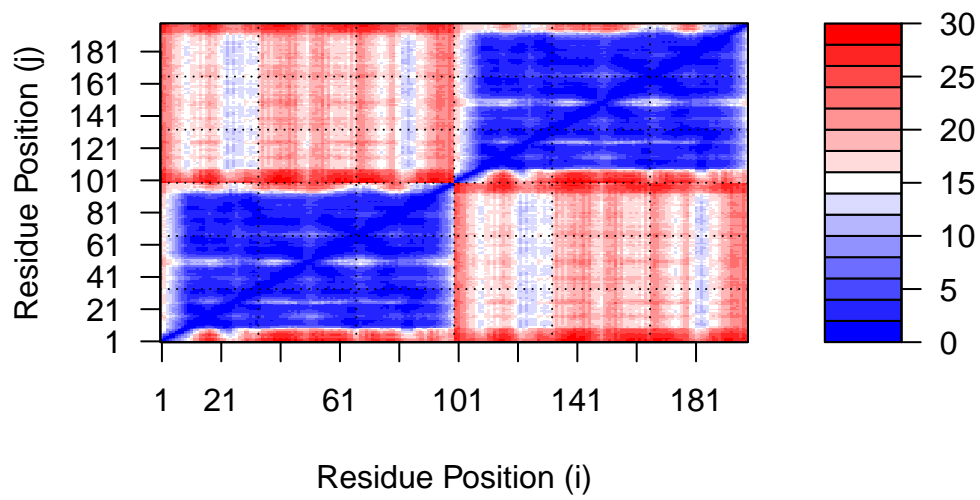
```
attributes(pae1)
```

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

```
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))
```

