Class 12: Protein Structure Prediction with AlphaFold

Alice Lai (PID:A16799081)

8. Custom analysis of resulting models

```
results_dir <- "hivprdimer_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] "dimer_23119_unrelaxed_rank_001_alphafold2_multimer_v3_model_1_seed_000.pdb"
[2] "dimer_23119_unrelaxed_rank_002_alphafold2_multimer_v3_model_5_seed_000.pdb"
[3] "dimer_23119_unrelaxed_rank_003_alphafold2_multimer_v3_model_4_seed_000.pdb"
[4] "dimer_23119_unrelaxed_rank_004_alphafold2_multimer_v3_model_2_seed_000.pdb"
[5] "dimer_23119_unrelaxed_rank_005_alphafold2_multimer_v3_model_3_seed_000.pdb"
```

```
library(bio3d)
#install.packages("BiocManager")
#BiocManager::install("msa")
```

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

```
Reading PDB files:
hivprdimer_23119//dimer_23119_unrelaxed_rank_001_alphafold2_multimer_v3_model_1_seed_000.pdb
hivprdimer_23119//dimer_23119_unrelaxed_rank_002_alphafold2_multimer_v3_model_5_seed_000.pdb
hivprdimer_23119//dimer_23119_unrelaxed_rank_003_alphafold2_multimer_v3_model_4_seed_000.pdb
hivprdimer_23119//dimer_23119_unrelaxed_rank_004_alphafold2_multimer_v3_model_2_seed_000.pdb
hivprdimer_23119//dimer_23119_unrelaxed_rank_005_alphafold2_multimer_v3_model_3_seed_000.pdb
.....

Extracting sequences

pdb/seq: 1   name: hivprdimer_23119//dimer_23119_unrelaxed_rank_001_alphafold2_multimer_v3_mo
pdb/seq: 2   name: hivprdimer_23119//dimer_23119_unrelaxed_rank_002_alphafold2_multimer_v3_mo
pdb/seq: 3   name: hivprdimer_23119//dimer_23119_unrelaxed_rank_003_alphafold2_multimer_v3_mo
pdb/seq: 4   name: hivprdimer_23119//dimer_23119_unrelaxed_rank_004_alphafold2_multimer_v3_mo
pdb/seq: 5   name: hivprdimer_23119//dimer_23119_unrelaxed_rank_005_alphafold2_multimer_v3_mo
```

```
pdbs
```

```
                              1        .         .         .         .        50
[Truncated_Name:1]dimer_2311   PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:2]dimer_2311   PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:3]dimer_2311   PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:4]dimer_2311   PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
[Truncated_Name:5]dimer_2311   PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGI
                               **************************************************
                              1        .         .         .         .        50

                             51        .         .         .         .       100
[Truncated_Name:1]dimer_2311   GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:2]dimer_2311   GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:3]dimer_2311   GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:4]dimer_2311   GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
[Truncated_Name:5]dimer_2311   GGFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNFP
                               **************************************************
                             51        .         .         .         .       100

                            101        .         .         .         .       150
[Truncated_Name:1]dimer_2311   QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIG
[Truncated_Name:2]dimer_2311   QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIG
[Truncated_Name:3]dimer_2311   QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIG
[Truncated_Name:4]dimer_2311   QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIG
```

```
[Truncated_Name:5]dimer_2311   QITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEMSLPGRWKPKMIGGIG
                               **************************************************
                              101        .         .         .         .         150

                              151        .         .         .         .       198
[Truncated_Name:1]dimer_2311   GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:2]dimer_2311   GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:3]dimer_2311   GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:4]dimer_2311   GFIKVRQYDQILIEICGHKAIGTVLVGPTPVNIIGRNLLTQIGCTLNF
[Truncated_Name:5]dimer_2311   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
```

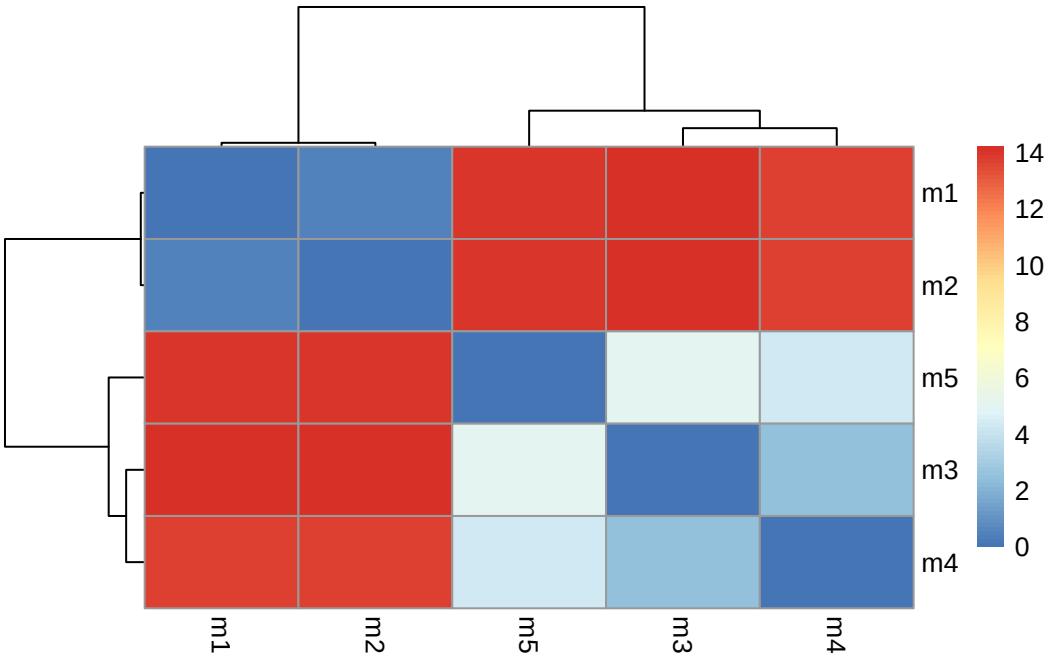
```
rd <- rmsd(pdbs, fit=T)
```

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

```
range(rd)
```

```
[1]  0.000 14.203
```

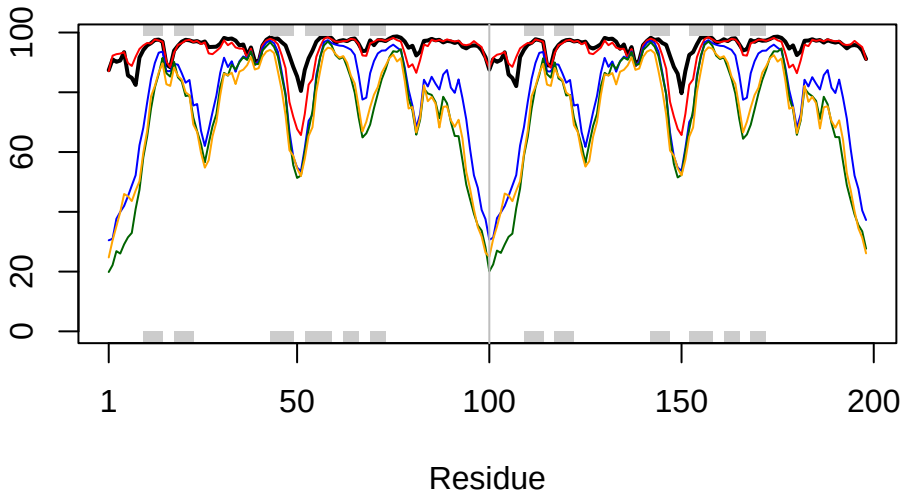
```
#install.packages("pheatmap")
library(pheatmap)

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(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")
```




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

```
core size 197 of 198  vol = 3124.961
core size 196 of 198  vol = 2920.851
core size 195 of 198  vol = 2759.61
core size 194 of 198  vol = 2621.721
core size 193 of 198  vol = 2506.06
core size 192 of 198  vol = 2431.57
core size 191 of 198  vol = 2389.645
core size 190 of 198  vol = 2361.021
core size 189 of 198  vol = 2369.16
core size 188 of 198  vol = 2363.057
core size 187 of 198  vol = 2376.959
core size 186 of 198  vol = 2355.91
core size 185 of 198  vol = 2346.191
core size 184 of 198  vol = 2306.353
core size 183 of 198  vol = 2255.869
core size 182 of 198  vol = 2190.649
core size 181 of 198  vol = 2116.308
core size 180 of 198  vol = 1992.733
core size 179 of 198  vol = 1949.969
core size 178 of 198  vol = 1893.838
```

```
core size 177 of 198  vol = 1829.766
core size 176 of 198  vol = 1752.857
core size 175 of 198  vol = 1678.022
core size 174 of 198  vol = 1604.938
core size 173 of 198  vol = 1562.773
core size 172 of 198  vol = 1530.882
core size 171 of 198  vol = 1484.989
core size 170 of 198  vol = 1440.135
core size 169 of 198  vol = 1398.99
core size 168 of 198  vol = 1357.399
core size 167 of 198  vol = 1314.429
core size 166 of 198  vol = 1266.817
core size 165 of 198  vol = 1225.544
core size 164 of 198  vol = 1188.065
core size 163 of 198  vol = 1156.619
core size 162 of 198  vol = 1091.277
core size 161 of 198  vol = 1058.55
core size 160 of 198  vol = 1010.855
core size 159 of 198  vol = 983.008
core size 158 of 198  vol = 957.153
core size 157 of 198  vol = 931.981
core size 156 of 198  vol = 905.995
core size 155 of 198  vol = 869.254
core size 154 of 198  vol = 843.978
core size 153 of 198  vol = 811.745
core size 152 of 198  vol = 783.826
core size 151 of 198  vol = 758.742
core size 150 of 198  vol = 728.577
core size 149 of 198  vol = 698.246
core size 148 of 198  vol = 678.143
core size 147 of 198  vol = 652.415
core size 146 of 198  vol = 636.917
core size 145 of 198  vol = 619.772
core size 144 of 198  vol = 603.091
core size 143 of 198  vol = 587.741
core size 142 of 198  vol = 573.444
core size 141 of 198  vol = 557.195
core size 140 of 198  vol = 539.015
core size 139 of 198  vol = 517.718
core size 138 of 198  vol = 496.614
core size 137 of 198  vol = 475.524
core size 136 of 198  vol = 459.064
core size 135 of 198  vol = 445.606
```

```
core size 134 of 198  vol = 430.657
core size 133 of 198  vol = 410.332
core size 132 of 198  vol = 401.044
core size 131 of 198  vol = 388.238
core size 130 of 198  vol = 375.86
core size 129 of 198  vol = 364.108
core size 128 of 198  vol = 350.533
core size 127 of 198  vol = 341.561
core size 126 of 198  vol = 328.474
core size 125 of 198  vol = 314.511
core size 124 of 198  vol = 300.607
core size 123 of 198  vol = 288.935
core size 122 of 198  vol = 277.27
core size 121 of 198  vol = 264.382
core size 120 of 198  vol = 254.352
core size 119 of 198  vol = 242.691
core size 118 of 198  vol = 231.991
core size 117 of 198  vol = 221.382
core size 116 of 198  vol = 212.788
core size 115 of 198  vol = 203.834
core size 114 of 198  vol = 194.898
core size 113 of 198  vol = 184.082
core size 112 of 198  vol = 172.93
core size 111 of 198  vol = 162.111
core size 110 of 198  vol = 151.154
core size 109 of 198  vol = 141.921
core size 108 of 198  vol = 131.714
core size 107 of 198  vol = 124.278
core size 106 of 198  vol = 118.708
core size 105 of 198  vol = 112.734
core size 104 of 198  vol = 106.464
core size 103 of 198  vol = 100.447
core size 102 of 198  vol = 92.93
core size 101 of 198  vol = 84.911
core size 100 of 198  vol = 77.129
core size 99 of 198  vol = 70.021
core size 98 of 198  vol = 62.159
core size 97 of 198  vol = 54.55
core size 96 of 198  vol = 47.345
core size 95 of 198  vol = 42.479
core size 94 of 198  vol = 37.149
core size 93 of 198  vol = 29.658
core size 92 of 198  vol = 22.749
```

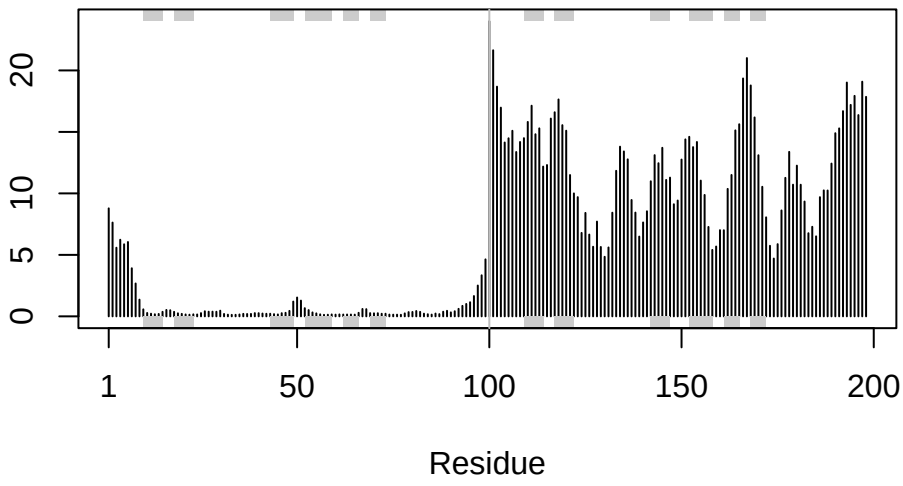
```
 core size 91 of 198  vol = 14.984
 core size 90 of 198  vol = 7.932
 core size 89 of 198  vol = 4.439
 core size 88 of 198  vol = 3.189
 core size 87 of 198  vol = 2.468
 core size 86 of 198  vol = 1.901
 core size 85 of 198  vol = 1.633
 core size 84 of 198  vol = 1.295
 core size 83 of 198  vol = 1.019
 core size 82 of 198  vol = 0.867
 core size 81 of 198  vol = 0.722
 core size 80 of 198  vol = 0.618
 core size 79 of 198  vol = 0.532
 core size 78 of 198  vol = 0.506
 core size 77 of 198  vol = 0.474
 FINISHED: Min vol ( 0.5 ) reached
```

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

```
# 78 positions (cumulative volume <= 0.5 Angstrom^3)
  start end length
1    10  48     39
2    53  66     14
3    68  92     25
```

```
xyz <- pdbfit(pdbs, 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] 87.38 91.00 90.19 90.62 93.44 85.62
```

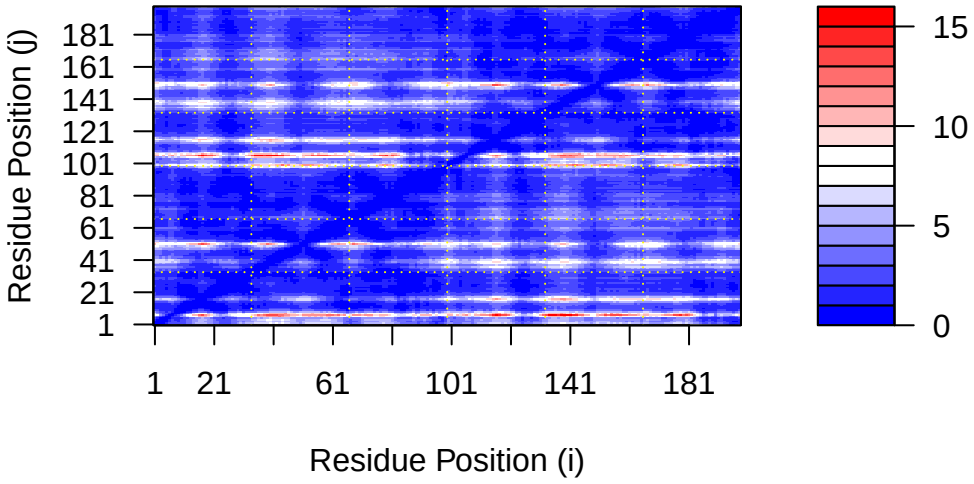
```
pae1$max_pae
```

```
[1] 15.875
```

```
pae5$max_pae
```

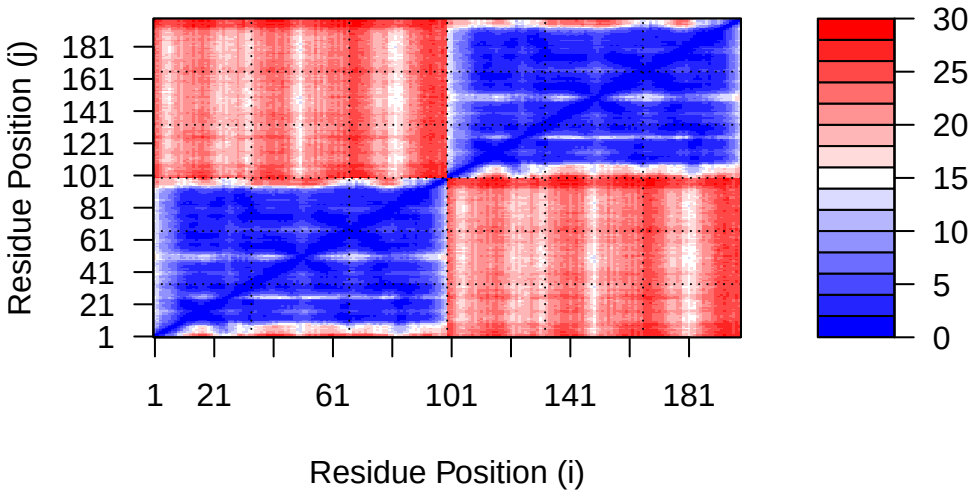
```
[1] 29.23438
```

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

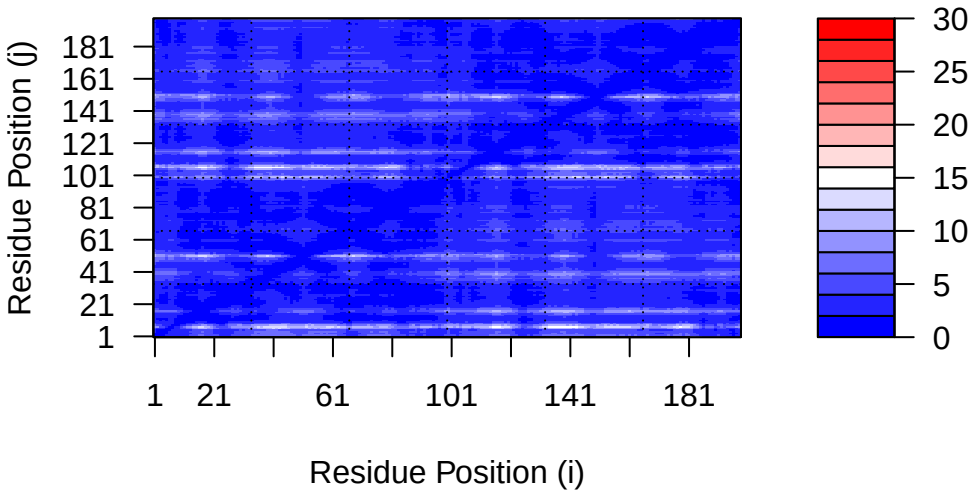



```
plot.dmat(pae5$pae,
          xlab="Residue Position (i)",
          ylab="Residue Position (j)",
          grid.col = "black",
```

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




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



Residue conservation from alignment file

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

```
[1] "hivprdimer_23119//dimer_23119.a3m"
```

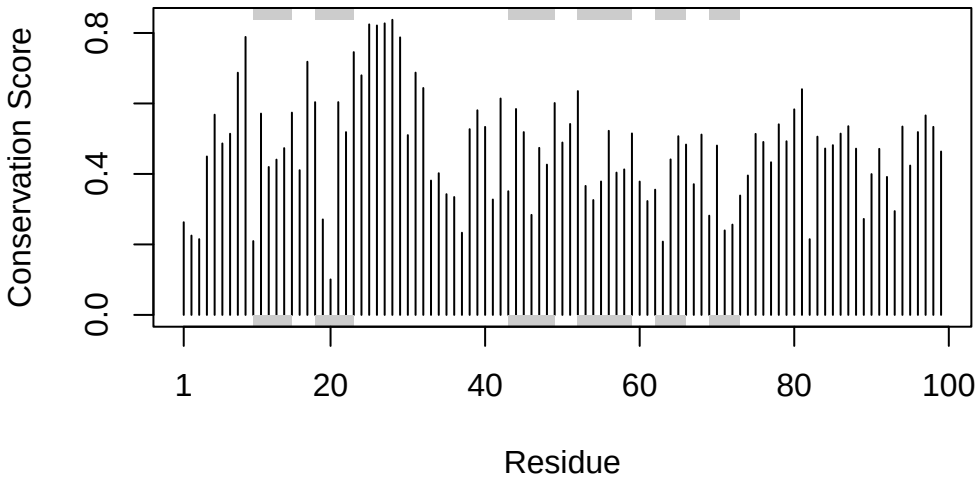
```
aln <- read.fasta(aln_file[1], to.upper = TRUE)
```

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

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

```
[1] 5378  132
```

```
sim <- conserv(aln)

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




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

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

```
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")
```