HW Class 6 (R Functions)

Blair Chang

```
library(bio3d)
s1 <- read.pdb("4AKE") # kinase with drug
```

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

```
s2 <- read.pdb("1AKE") # kinase no drug
```

```
Note: Accessing on-line PDB file
 PDB has ALT records, taking A only, rm.alt=TRUE
```

```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

```
s1
```

```
Call:  read.pdb(file = "4AKE")

  Total Models#: 1
    Total Atoms#: 3459,  XYZs#: 10377  Chains#: 2  (values: A B)

    Protein Atoms#: 3312  (residues/Calpha atoms#: 428)
    Nucleic acid Atoms#: 0  (residues/phosphate atoms#: 0)

    Non-protein/nucleic Atoms#: 147  (residues: 147)
```

```
     Non-protein/nucleic resid values: [ HOH (147) ]

   Protein sequence:
      MRIILLGAPGAGKGTQAQFIMEKYGIPQISTGDMLRAAVKSGSELGKQAKDIMDAGKLVT
      DELVIALVKERIAQEDCRNGFLLDGFPRTIPQADAMKEAGINVDYVLEFDVPDELIVDRI
      VGRRVHAPSGRVYHVKFNPPKVEGKDDVTGEELTTRKDDQEETVRKRLVEYHQMTAPLIG
      YYSKEAEAGNTKYAKVDGTKPVAEVRADLEKILGMRIILLGAPGA...<cut>...KILG

+ attr: atom, xyz, seqres, helix, sheet,
        calpha, remark, call
```

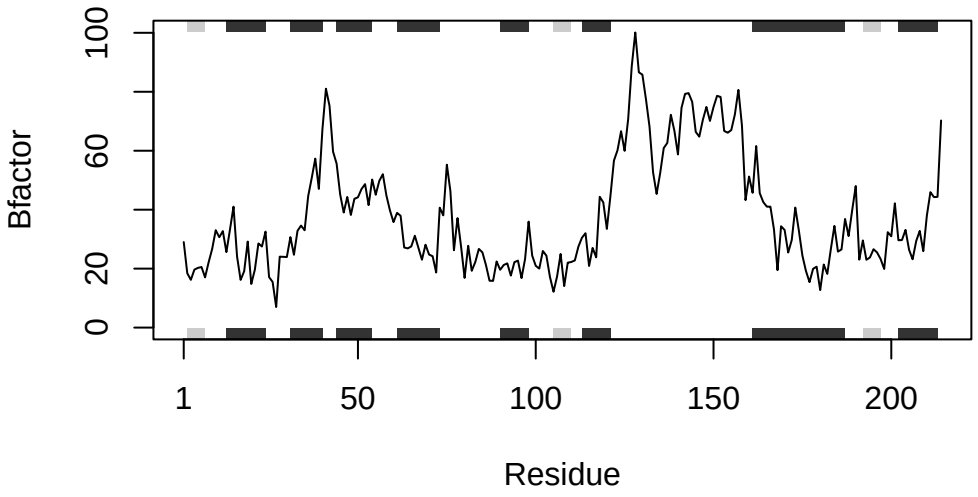
Q1 read.pdb will read a Protein Data Bank file and give us info such as Protein sequence, and total atoms.

```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")

s1.chainA
```

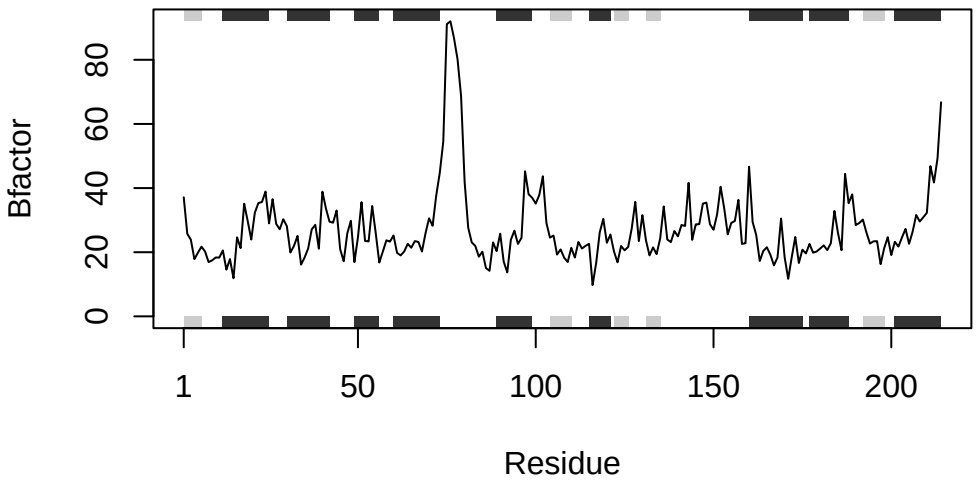
```
 Call:  trim.pdb(pdb = s1, chain = "A", elety = "CA")

   Total Models#: 1
     Total Atoms#: 214,  XYZs#: 642  Chains#: 1  (values: A)

     Protein Atoms#: 214  (residues/Calpha atoms#: 214)
     Nucleic acid Atoms#: 0  (residues/phosphate atoms#: 0)

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

   Protein sequence:
      MRIILLGAPGAGKGTQAQFIMEKYGIPQISTGDMLRAAVKSGSELGKQAKDIMDAGKLVT
      DELVIALVKERIAQEDCRNGFLLDGFPRTIPQADAMKEAGINVDYVLEFDVPDELIVDRI
      VGRRVHAPSGRVYHVKFNPPKVEGKDDVTGEELTTRKDDQEETVRKRLVEYHQMTAPLIG
      YYSKEAEAGNTKYAKVDGTKPVAEVRADLEKILG

+ attr: atom, helix, sheet, seqres, xyz,
        calpha, call
```

Q2 trim.pdb will give us a new smaller PDB object from the larger PDB object.

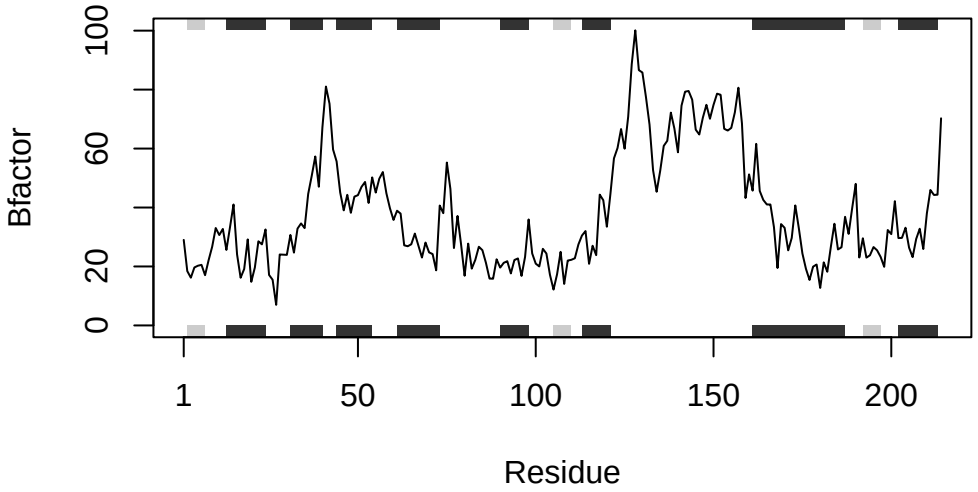
```
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b

plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



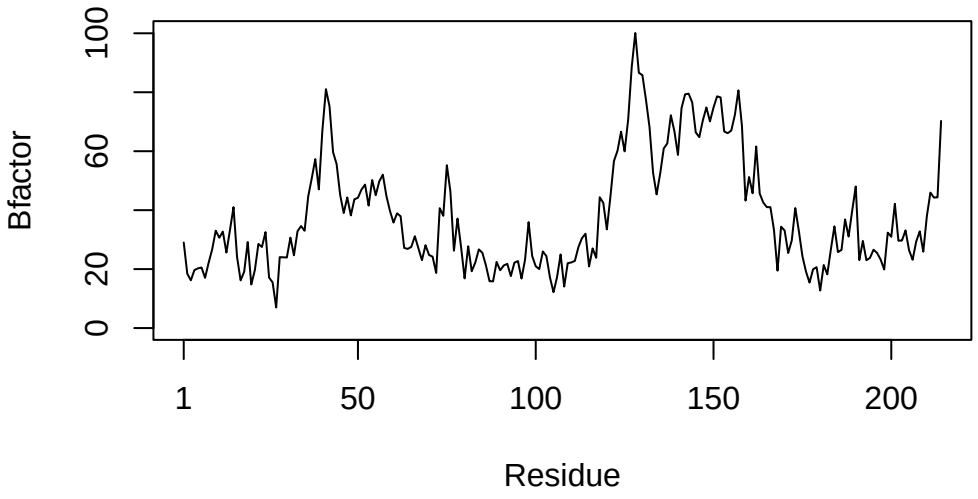

```
plotb3(s2.b, sse=s2.chainA, typ="l", ylab="Bfactor")
```




```
plotb3(s3.b, sse=s3.chainA, typ="l", ylab="Bfactor")
```



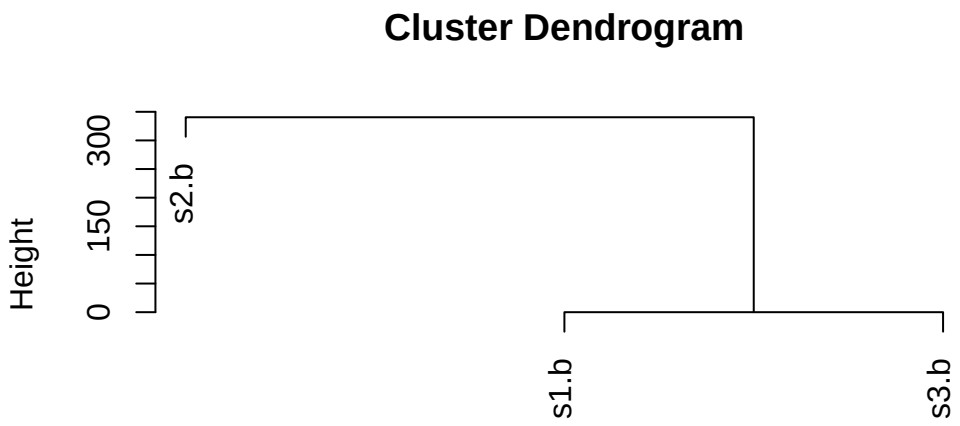
```
plotb3(s3.b, sse=NULL, typ="l", ylab="Bfactor")
```



Q3 sse = NULL will turn off the marginal black and grey rectangles. They represent the secondary structure object as returned from Secondary Structure Analysis with DSSP or STRIDE.

Q4 ggplot or Hierarchical Clustering (hclustplot)

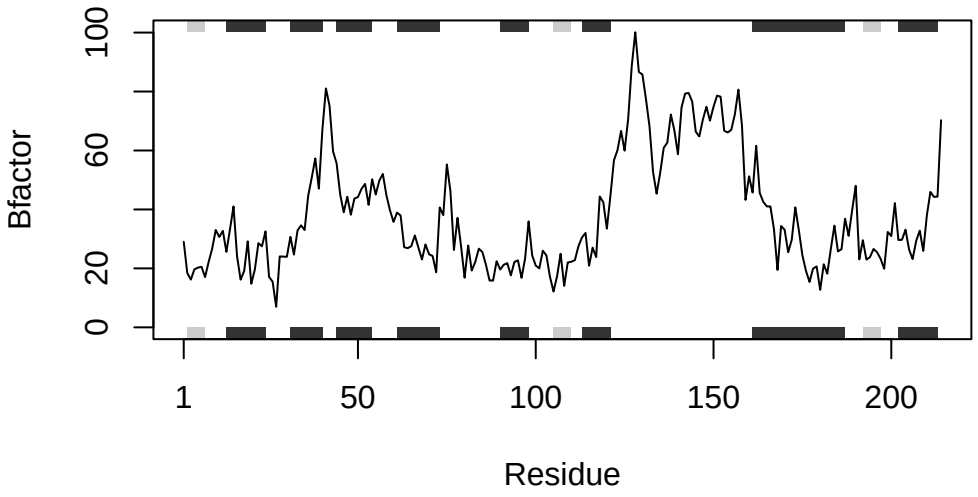
```
hc <- hclust( dist( rbind(s1.b, s2.b, s3.b) ) )
plot(hc)
```



Q5 s1.b and s3.b

Q6 How would you generalize the original code above to work with any set of input protein structures?

```
Function_Protein_Analysis <- function(x){
  plotb3(trim.pdb(x, chain="A", elety="CA")$atom$b,
         sse = trim.pdb(x, chain="A", elety="CA"),
         typ="l", ylab = "Bfactor")}

Function_Protein_Analysis(s1)
```




```
#compare my function plot to the original plot. SAME!
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```