# HW Class 6

Linh Tran (PID: A16435846)

2024-04-27

## Generalize a code to work with any set of input protein structures

First, I need to install `bio3d` using `install.packages("bio3d")` in console and use `library()` to retrieve it.

```
library(bio3d)
```

### Analysis code

First, the PDB files of proteins can be retrieved using `read.pdb()`

For example, the 3 proteins of interest can be retrieved

```
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
```

Next, we can use `trim.pdb()` to filter out structures from the PDBs. In this case, we want to get chain A

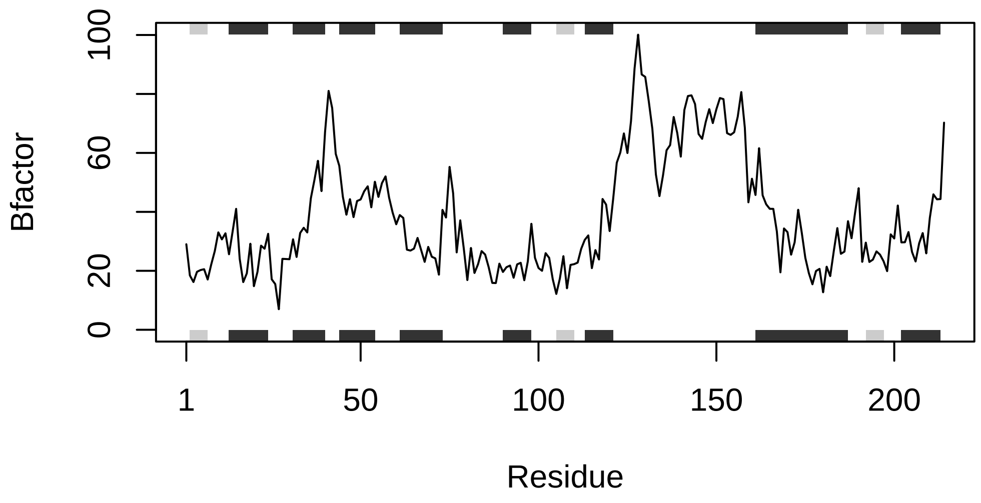
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")
```

Then, to get B-factor data from the atomic coordinate ATOM data of each protein, we can specify with `$atom$b`

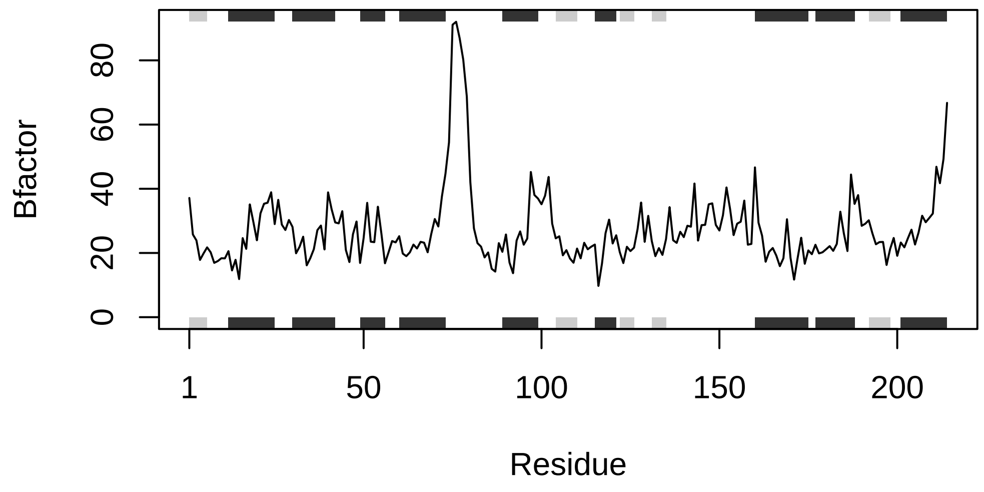
```
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
```

Lastly, we can use `plotb3()` to plot the B-factor trends. The type of plot is specified by `typ="l"` to produce a line plot.

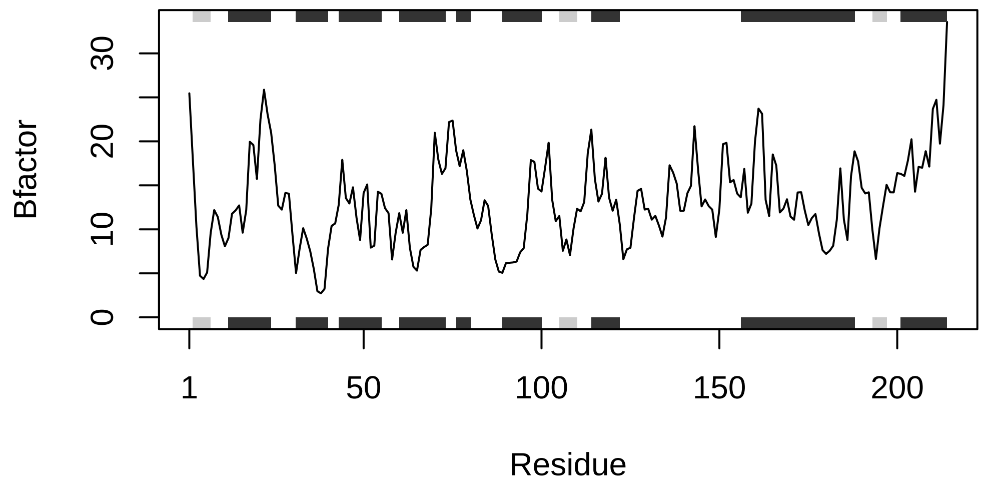
```
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



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

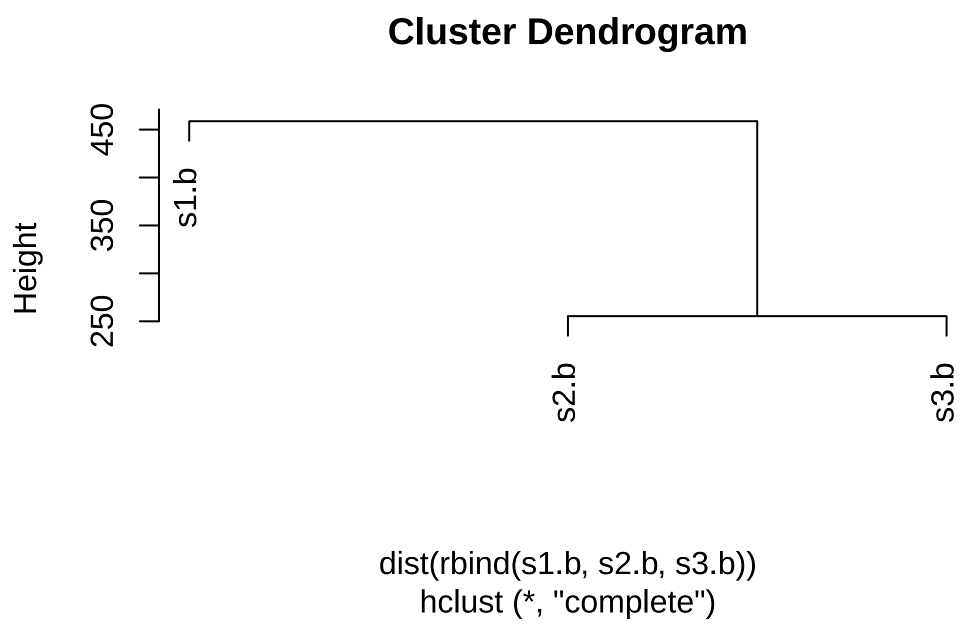


```
plotb3(s3.b, sse=s3.chainA, typ="l", ylab="Bfactor")
```



Additionally, a cluster dendrogram can be made to show the proteins that are more similar to one another based on their B-factor trends.

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



> Q6. Write your own function starting from the code above that analyzes protein drug interactions by reading in any protein PDB data and outputs a plot for the specified protein.

In order to generalize a code that would work with any set of input protein structures, each protein must be retrieved with `read.pdb()` and stored as `p#` to be inputted into the a generalized code made by using `function()` to combine the codes above.

```
p1 <- read.pdb("4AKE")
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/9d/fhz0yltj5yd2yqd_p9zyvw0w0000gn/T//RtmpaiIfwc/4AKE.pdb exists.
Skipping download
```

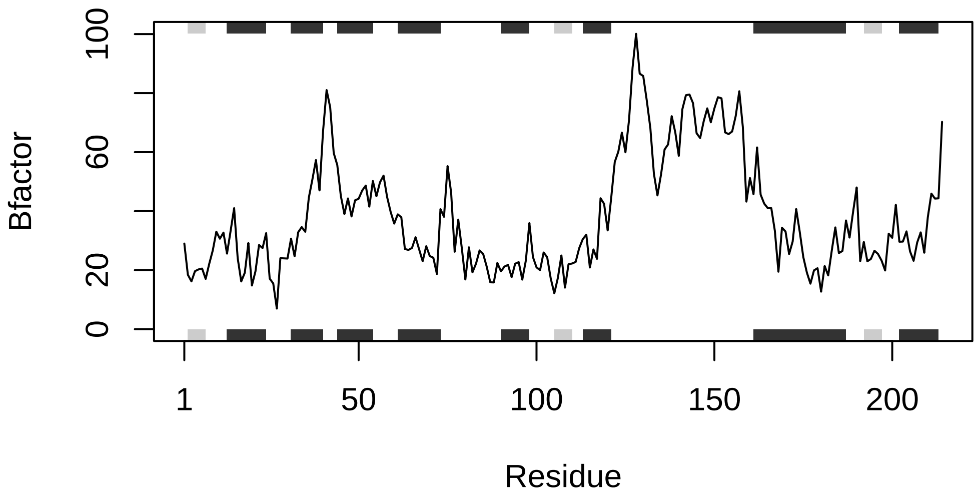
```r
p2 <- read.pdb("1AKE")
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/9d/fhz0yltj5yd2yqd_p9zyvw0w0000gn/T//RtmpaiIfwc/1AKE.pdb exists.
Skipping download

   PDB has ALT records, taking A only, rm.alt=TRUE
```

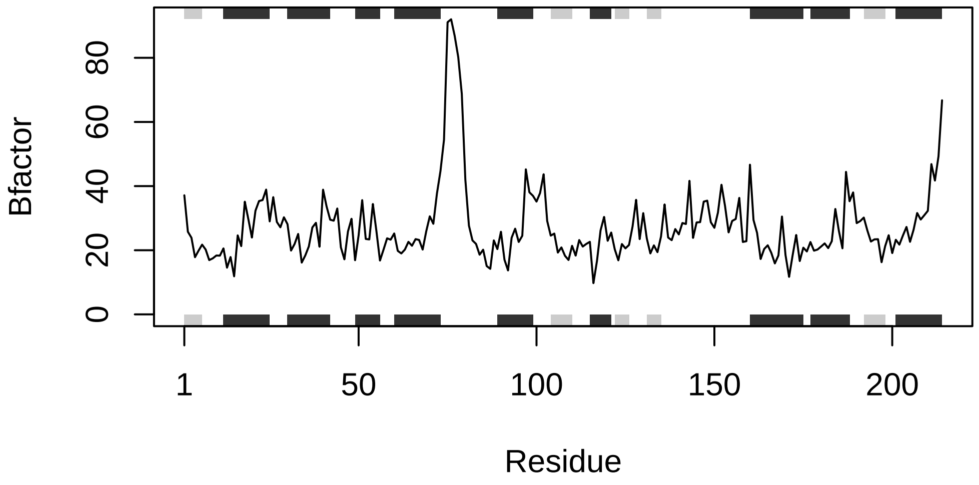
```r
p3 <- read.pdb("1E4Y")
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/9d/fhz0yltj5yd2yqd_p9zyvw0w0000gn/T//RtmpaiIfwc/1E4Y.pdb exists.
Skipping download
```

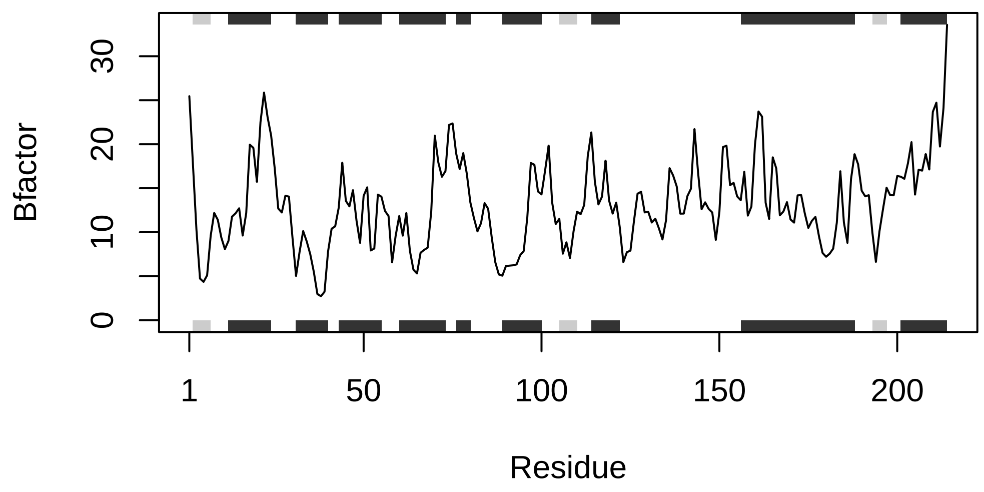
```r
pdb_ <- function(x){protein.chainA <- trim.pdb(x, chain="A", elety="CA")
  protein.b <- protein.chainA$atom$b
  return(plotb3(protein.b, sse=protein.chainA, typ="l", ylab="Bfactor"))
  }
pdb_(p1)
```


Bfactor
0
20
60
100
1
50
100
150
200
Residue

```
pdb_(p2)
```


Bfactor
0
20
40
60
80
1
50
100
150
200
Residue

```
pdb_(p3)
```


Bfactor
Residue