HW06

Rachel Felix, PID: A16037641

Section 1: Improving analysis code by writing functions

A. Can you improve this analysis code?

```
df <- data.frame(a=1:10, b=seq(200,400,length=10),c=11:20,d=NA)
df$a <- (df$a - min(df$a)) / (max(df$a) - min(df$a))
df$b <- (df$b - min(df$a)) / (max(df$b) - min(df$b))
df$c <- (df$c - min(df$c)) / (max(df$c) - min(df$c))
df$d <- (df$d - min(df$d)) / (max(df$a) - min(df$d))
```

- Running through the code to see if it works:

```
df$a
```

```
 [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
 [8] 0.7777778 0.8888889 1.0000000
```

```
df$b
```

```
 [1] 1.000000 1.111111 1.222222 1.333333 1.444444 1.555556 1.666667 1.777778
 [9] 1.888889 2.000000
```

```
df$c
```

```
 [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
 [8] 0.7777778 0.8888889 1.0000000
```

```
df$d
```

```
 [1] NA NA NA NA NA NA NA NA NA NA
```

Correcting data frame

- Correct Min-Max Scaling for column 'a'

```
df$a <- (df$a - min(df$a)) / (max(df$a) - min(df$a))
```

- Correct Min-Max Scaling for column 'b'

```
df$b <- (df$b - min(df$b)) / (max(df$b) - min(df$b))
```

- Correct Min-Max Scaling for column 'c'

```
df$c <- (df$c - min(df$c)) / (max(df$c) - min(df$c))
```

- Ignore column d, since all values are NA

Simplifying to a core working code snippet

```
min_max_scale <- function(x) {
  (x - min(x, na.rm=FALSE)) / (max(x, na.rm=FALSE) - min(x, na.rm=FALSE))
}
```

```
lapply(df, min_max_scale)
```

```
$a
 [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
 [8] 0.7777778 0.8888889 1.0000000

$b
 [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
 [8] 0.7777778 0.8888889 1.0000000

$c
```

```
 [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
 [8] 0.7777778 0.8888889 1.0000000

$d
 [1] NA NA NA NA NA NA NA NA NA NA
```

B. Next improve the below example code for the analysis of protein drug interactions

- Original code

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

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

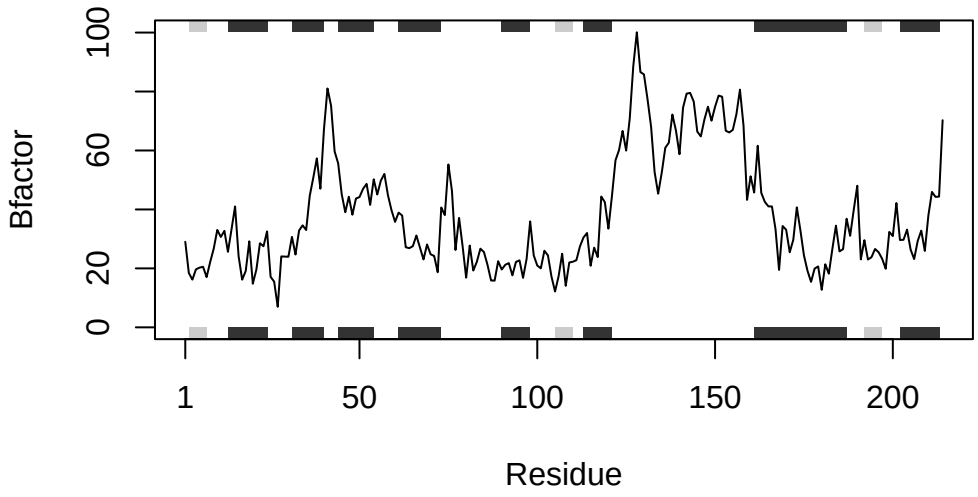
```
s2 <- read.pdb("1AKE") # kinase no drug
```

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

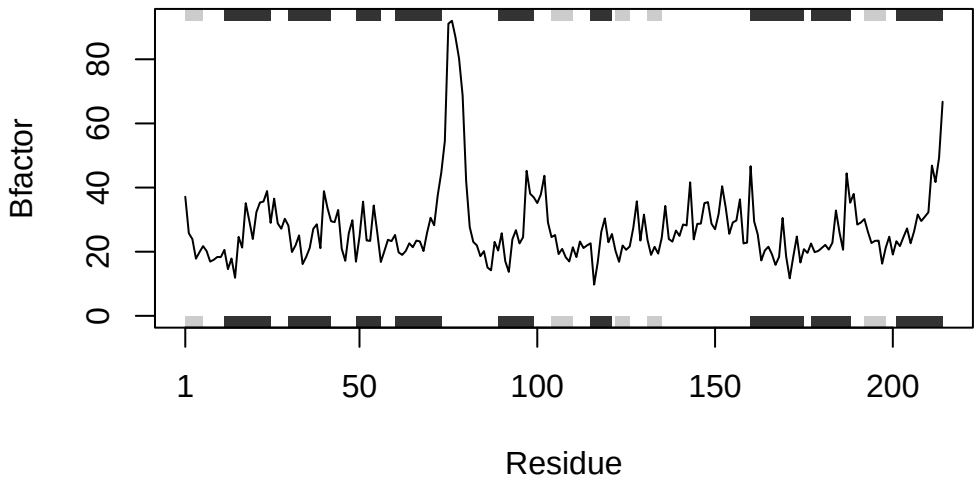
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

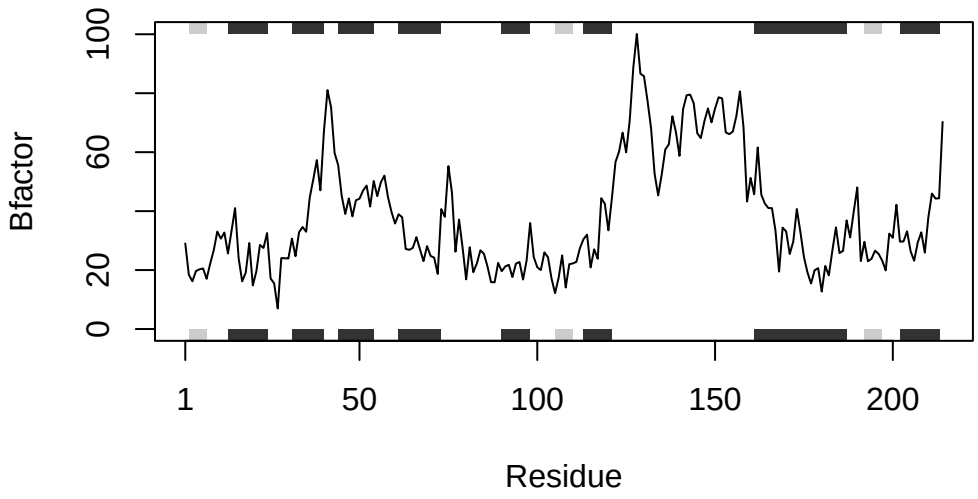
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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



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

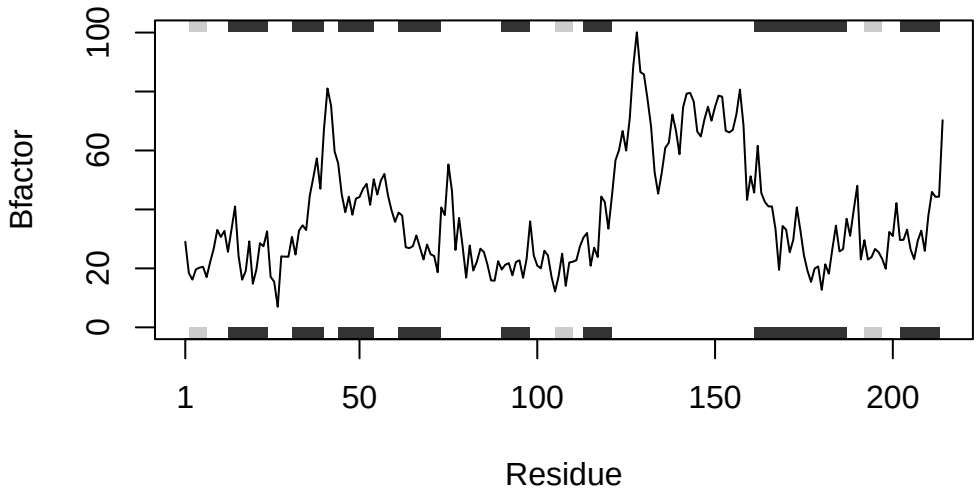


- Improved Code

```
library(bio3d)
plot_bfactor <- function(pdb_id, chain="A", elety="CA") {
  pdb <- read.pdb(pdb_id)
  trimmed <- trim.pdb(pdb, chain=chain, elety=elety)
  b_factors <- trimmed$atom$b
  plotb3(b_factors, sse=trimmed, typ="l", ylab="Bfactor")
}
plot_bfactor("4AKE")
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/46/s1zr4cld6l9gnt7h6p8xt4km0000gn/T//RtmpORTzNh/4AKE.pdb exists.
Skipping download
```

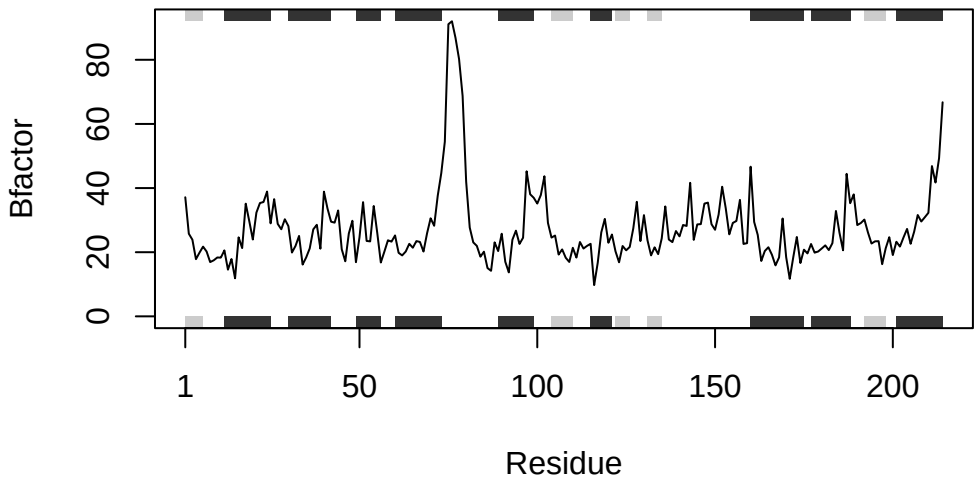



```
plot_bfactor("1AKE")
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/46/s1zr4cld6l9gnt7h6p8xt4km0000gn/T//RtmpORTzNh/1AKE.pdb exists.
Skipping download
```

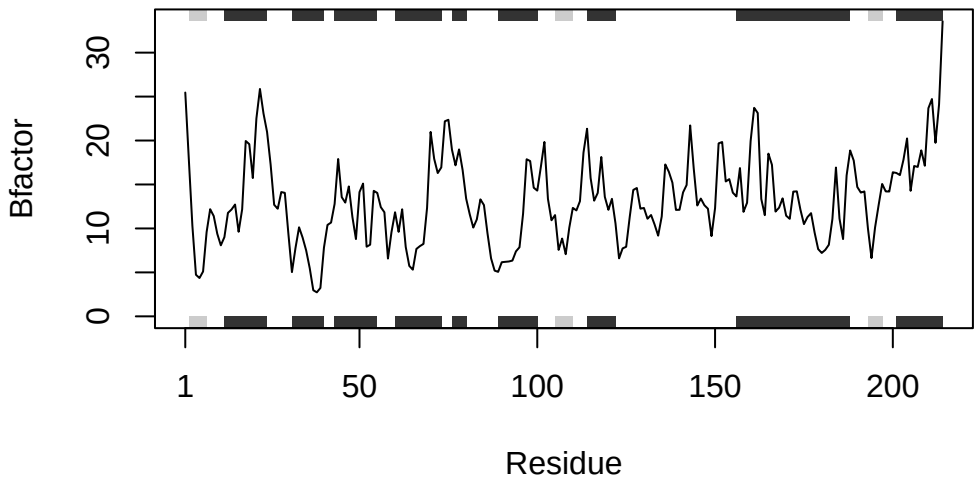
```
   PDB has ALT records, taking A only, rm.alt=TRUE
```




```
plot_bfactor("1E4Y")
```

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

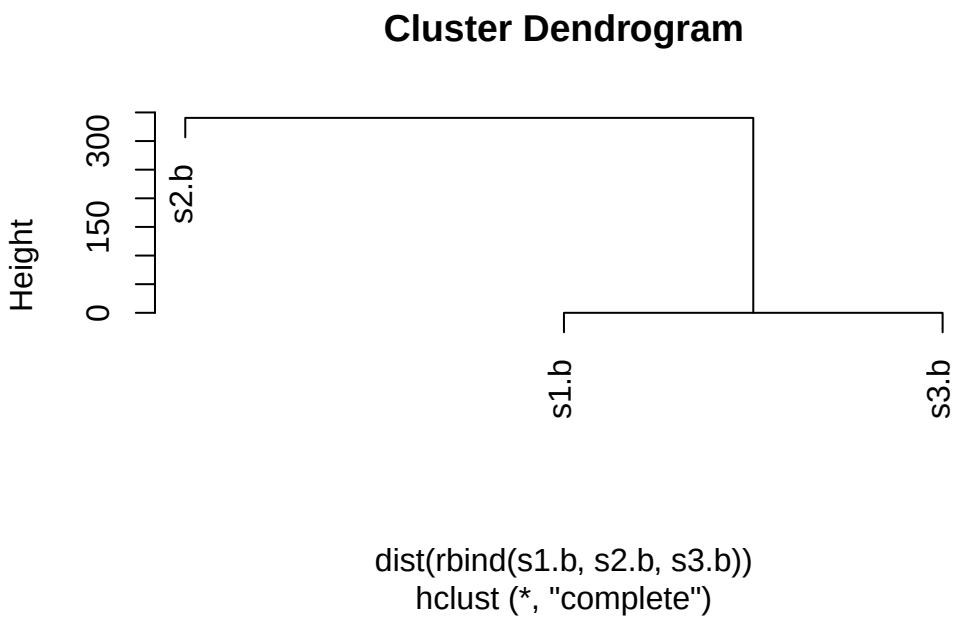
```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/46/s1zr4cld6l9gnt7h6p8xt4km0000gn/T//RtmpORTzNh/1E4Y.pdb exists.
Skipping download
```



Homework Questions

- Q1. What type of object is returned from the read.pdb() function? The `read.pdb()` function returns an object from the PDB file that contains atomic coordinates, annotation data, and other structure information.
- Q2. What does the trim.pdb() function do? The trim.pdb() function extracts specific chains and atoms for further analysis from the PDB object
- Q3. What input parameter would turn off the marginal black and grey rectangles in the plots and what do they represent in this case? Turning the function `sse=trimmed` instead to `sse=FALSE` would turn off the marginal black and grey rectangles, which in this case represent secondary structure elements (black for helices, grey for strands) in the protein structure
- Q4. What would be a better plot to compare across the different proteins? To compare across the proteins it might be helpful to plot all proteins on one graph using a specific color for each protein. That way you can more easily see B-factor profiles for Calpha atoms in chain A of each protein, where there are different regions of higher or lower flexibility along the sequence.
- Q5. Which proteins are more similar to each other in their B-factor trends. How could you quantify this? We see through the code below, that s1.b and s3.b are more similar to each other in their B-factor trends.

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



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

```
cluster_bfactors <- function(bfactor_list) {
  b_matrix <- do.call(rbind, bfactor_list)
  hc <- hclust(dist(b_matrix))
  plot(hc)
}
```