Class6 HW

Winnie Zhou (A16673200)

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))
```

Copy and paste errors

```
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$b - min(df$b)) / (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$d) - min(df$d))
```

Use working snippet of the code

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

Reduce calculation duplication

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

Use `range()` function

```
rng <- range(df$a)
x <- (df$a - rng[1]) / (rng[2] - rng[1])
```

Turn it into a function with "name", "arguments" and "body"

```
rescale <- function(x){rng <- range(df$a)
(df$a - rng[1]) / (rng[2] - rng[1])}
```

Test on small sample

```
rescale(1:10)
```

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

Test if function works for c(1,2,NA,3,10)

```
rescale <- function(df, na.rm=TRUE){
  rng <- range(df, na.rm=TRUE)
(df - rng[1]) / (rng[2] - rng[1])}
```

```
rescale(c(1,2,NA,3,10))
```

```
[1] 0.0000000 0.1111111        NA 0.2222222 1.0000000
```

B. Can you improve this analysis code?

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

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

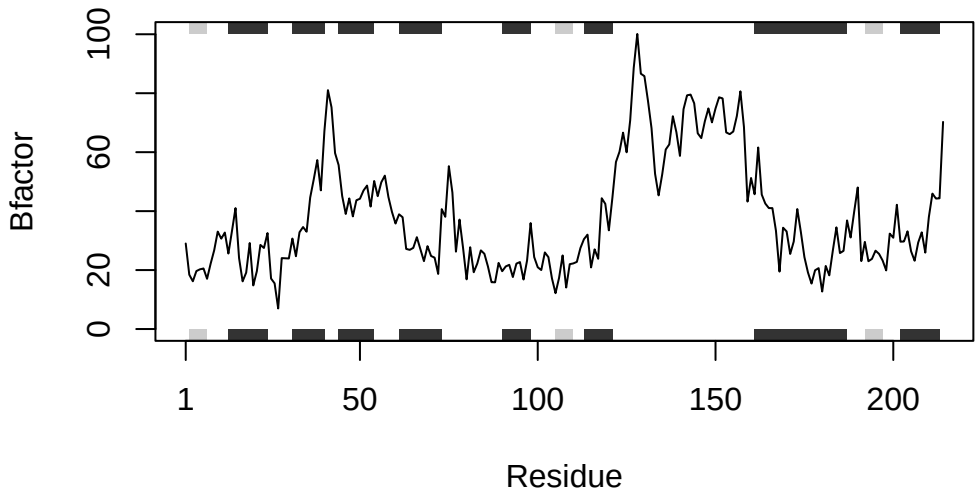
```
s2 <- read.pdb("1AKE") # kinase no drug
```

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

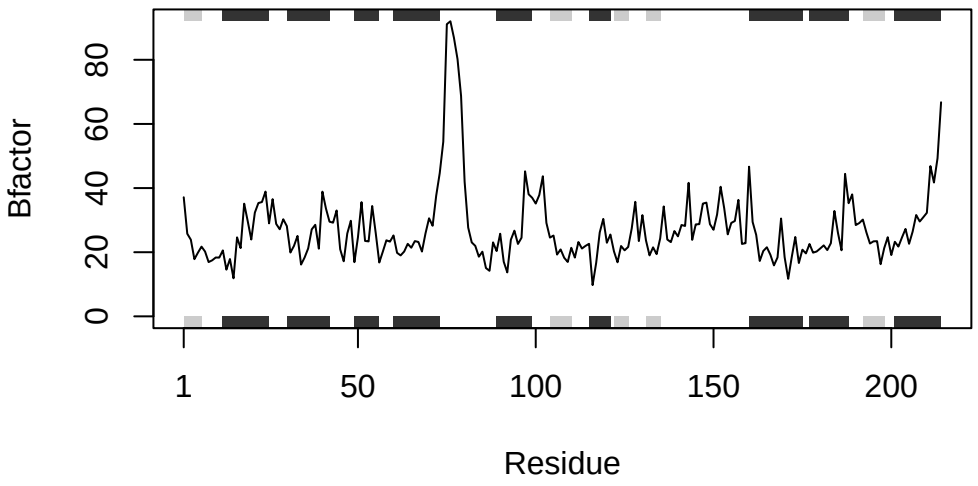
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

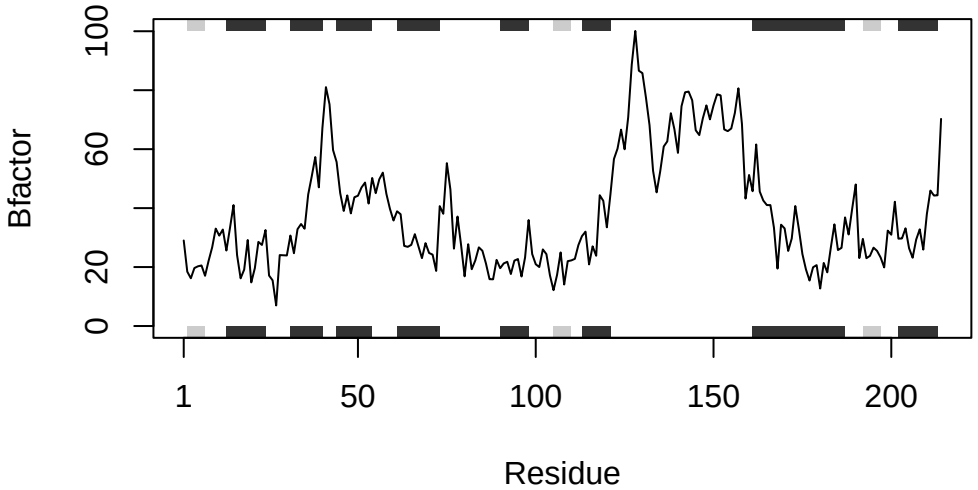
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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




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



Fixing errors: changes “s1” to “s3” in line 89

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

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/1y/t8s3xgz97rg_fclq5rv4c1tw0000gn/T//RtmpQ1uAu2/4AKE.pdb exists.
Skipping download
```

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

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/1y/t8s3xgz97rg_fclq5rv4c1tw0000gn/T//RtmpQ1uAu2/1AKE.pdb exists.
Skipping download
```

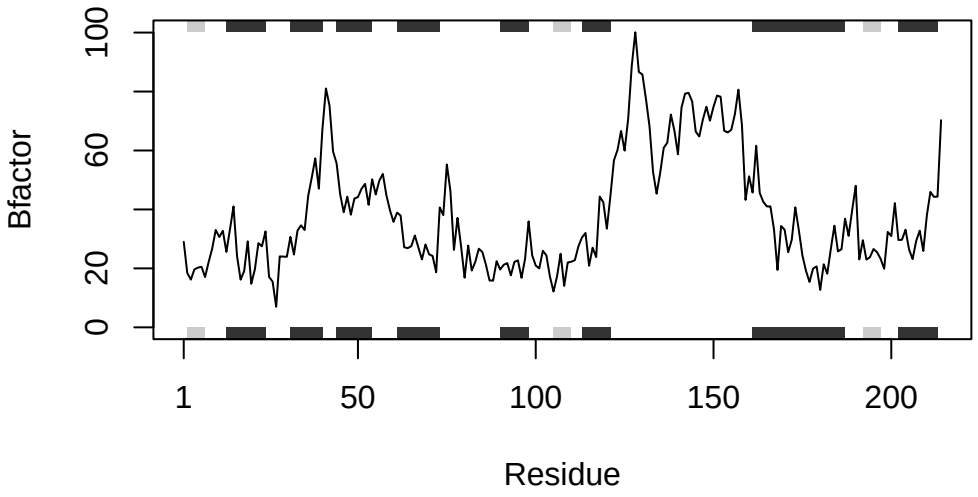
```
   PDB has ALT records, taking A only, rm.alt=TRUE
```

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

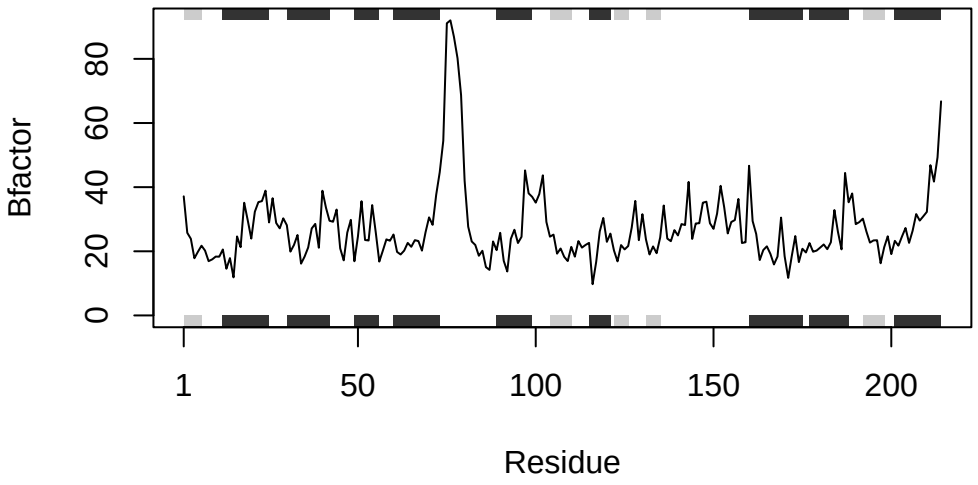
```
  Note: Accessing on-line PDB file
```

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/1y/t8s3xgz97rg_fclq5rv4c1tw0000gn/T//RtmpQ1uAu2/1E4Y.pdb exists.
Skipping download
```

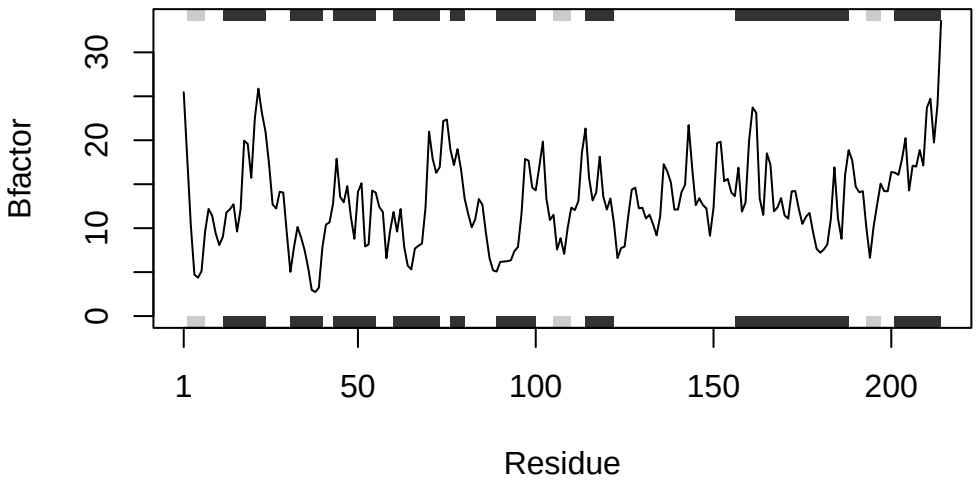
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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



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



Q1. What type of object is returned from the read.pdb() function?

`read.pdb()` function allows R to read a Protein Data Bank file which contains biomolecular data.

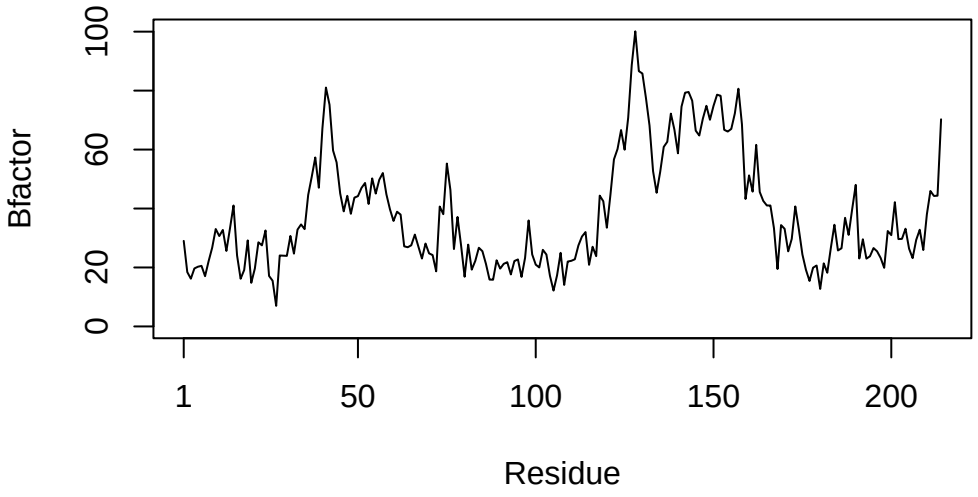
Q2. What does the trim.pdb() function do?

`trim.pdb()` function removes data that is not necessary for our analyses or visualization of structures, such as specific chains.

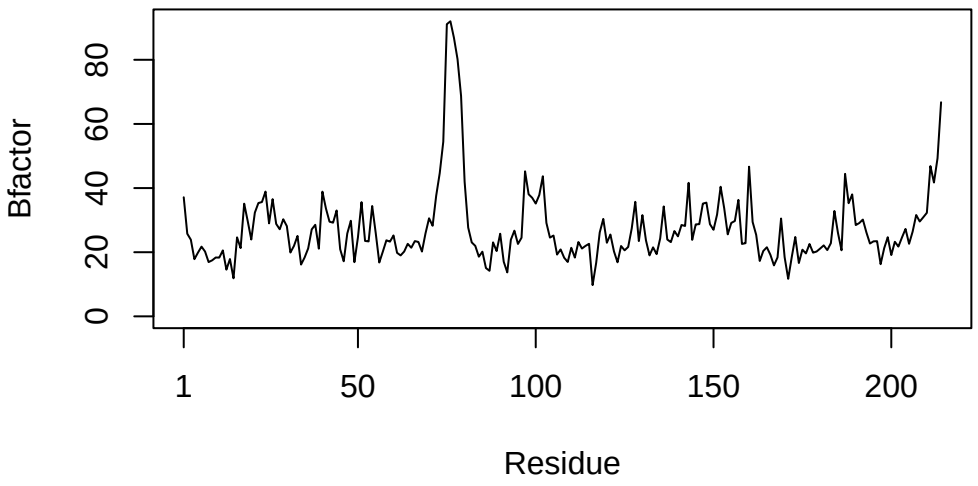
Q3. What input parameter would turn off the marginal black and grey rectangles in the plots and what do they represent in this case?

The marginal black and grey rectangles in the plots represent the 'classic' form of secondary structure annotation. We can turn them off by specifying the arguments: top=FALSE, bottom=FALSE.

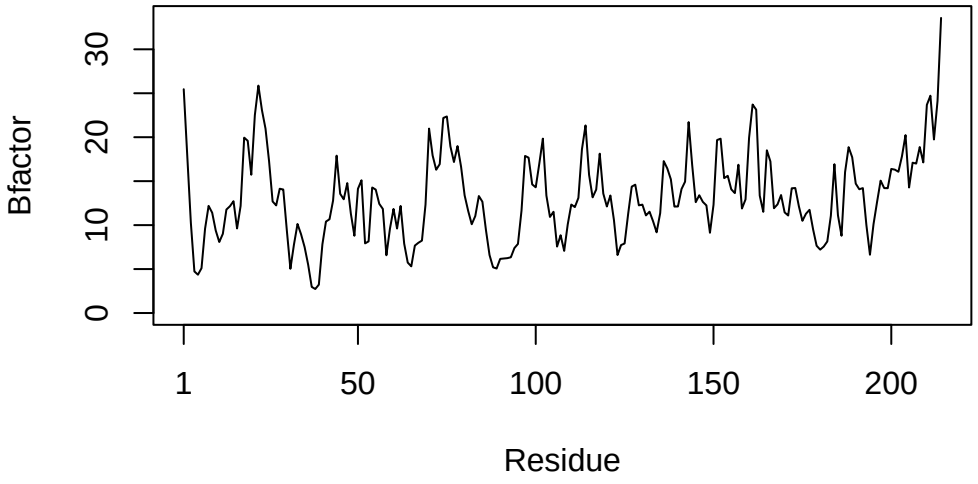
```
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor", top=FALSE, bot=FALSE)
```




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

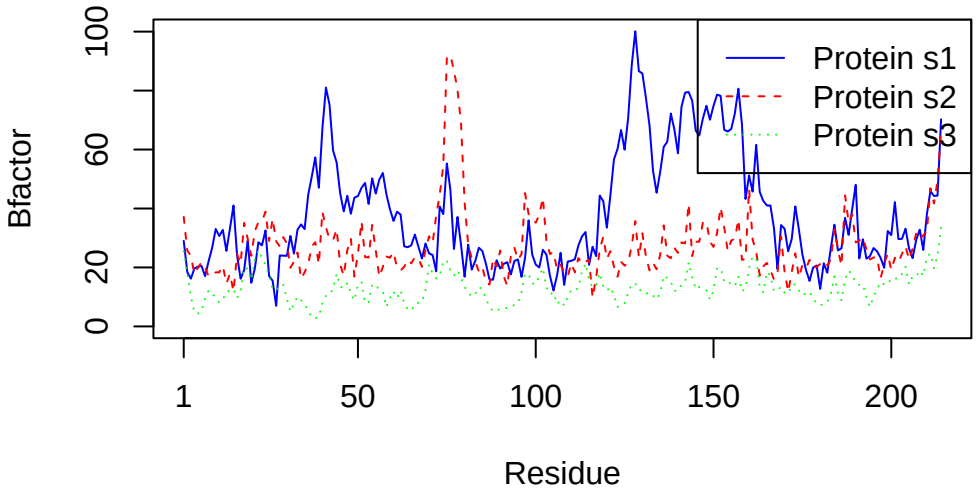



```
plotb3(s3.b, sse=s3.chainA, typ="l", ylab="Bfactor", top=FALSE, bot=FALSE)
```



Q4. What would be a better plot to compare across the different proteins?

```
plotb3(s1.b, sse = s1.chainA, typ = "l", ylab = "Bfactor", top = FALSE, bot = FALSE, col =
lines(s2.b, col = "red", lty = 2)
lines(s3.b, col = "green", lty = 3)
legend("topright", legend = c("Protein s1", "Protein s2", "Protein s3"), col = c("blue", "
```



Q5. Which proteins are more similar to each other in their B-factor trends? How could you quantify this?

We can quantify how similar the proteins are to each other by calculating the pearson correlaation coefficients. Protein s2 and s3 are the most similar to each other given they have the highest correlation: 0.5484932. We can also use hierarchial clustering. The proteins closest to each other, best not separated by a vertical branch, are more similar to each other.

```
correlation_s1_s2 <- cor(s1.b, s2.b)
correlation_s1_s3 <- cor(s1.b, s3.b)
correlation_s2_s3 <- cor(s2.b, s3.b)
correlation <- c(correlation_s1_s2, correlation_s1_s3, correlation_s2_s3)
highest <- max(correlation)
```

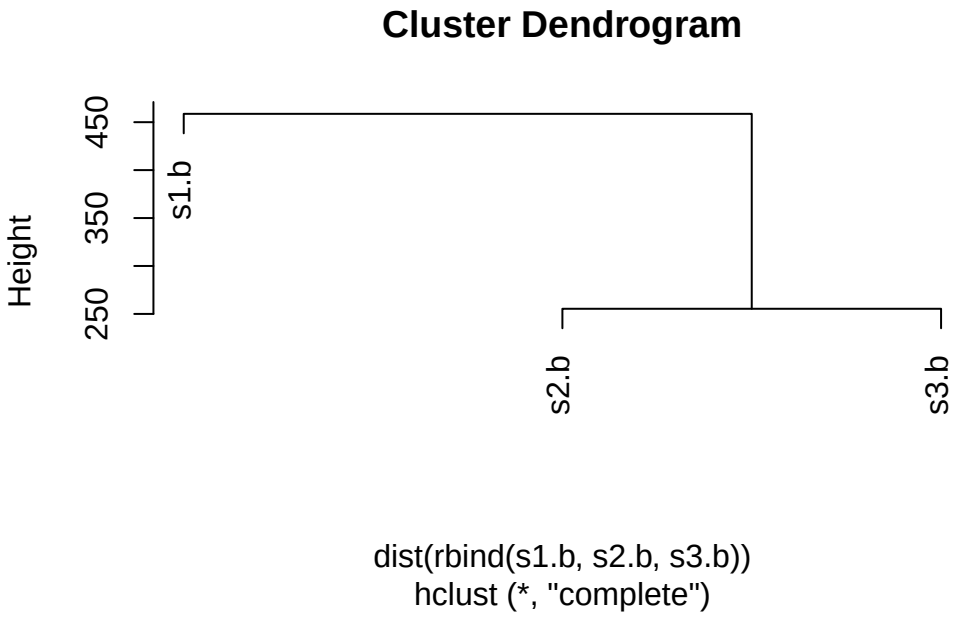
```
highest
```

```
[1] 0.5484932
```

```
which.max(correlation)
```

```
[1] 3
```

```
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?

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

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/1y/t8s3xgz97rg_fclq5rv4c1tw0000gn/T//RtmpQ1uAu2/4AKE.pdb exists.
Skipping download
```

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

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/1y/t8s3xgz97rg_fclq5rv4c1tw0000gn/T//RtmpQ1uAu2/1AKE.pdb exists.
Skipping download
```

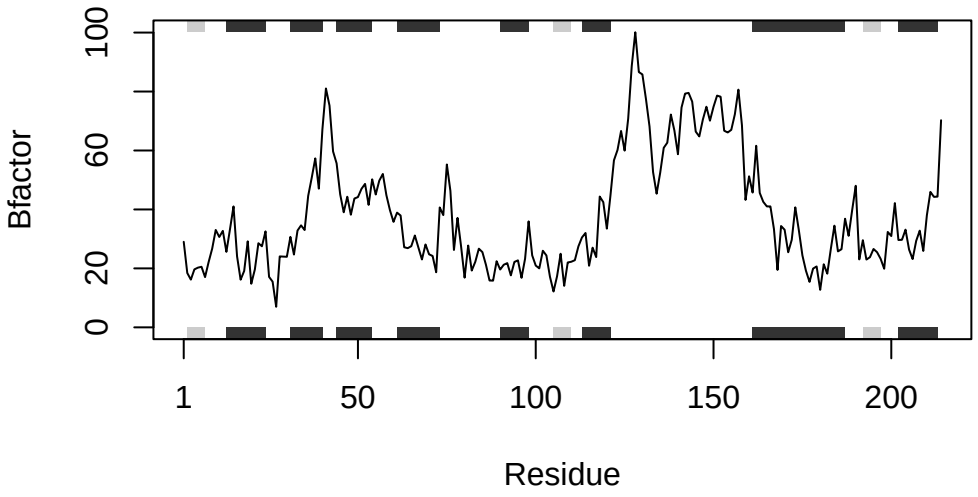
```
   PDB has ALT records, taking A only, rm.alt=TRUE
```

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

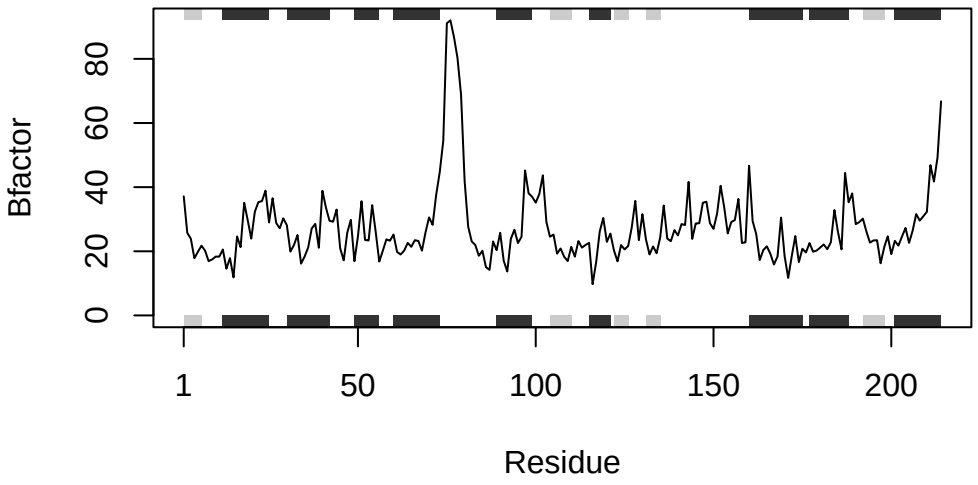
```
  Note: Accessing on-line PDB file
```

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/1y/t8s3xgz97rg_fclq5rv4c1tw0000gn/T//RtmpQ1uAu2/1E4Y.pdb exists.
Skipping download
```

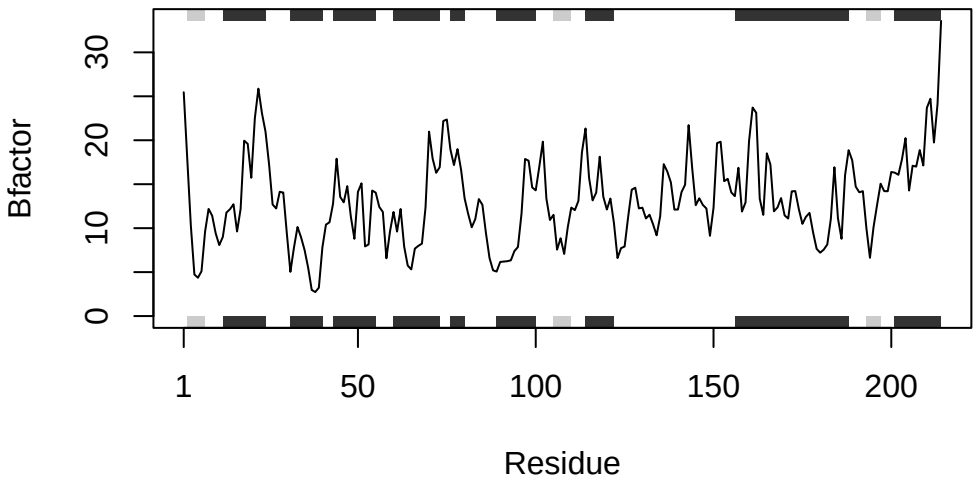
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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



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



To make sure this works with any set of input protein structures, we can substitute the pdb files with a generic variable (x) into a working snippet.

```
library(bio3d)
# We first set generic variable "x" as "4AKE" as an example usage
x <- "4AKE"

# x can be replaced for any set of input structure and we assign the `read.pdb()` function
structure <- read.pdb(x)
```

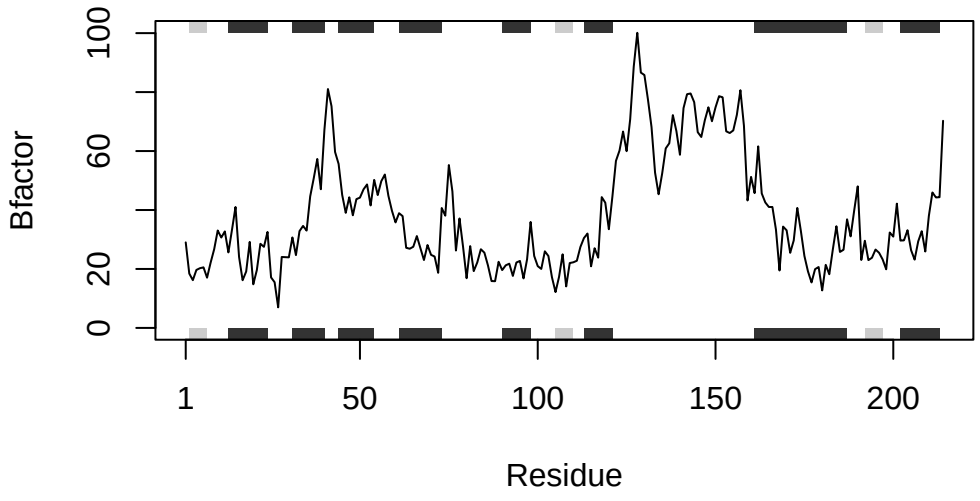
```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/1y/t8s3xgz97rg_fclq5rv4c1tw0000gn/T//RtmpQ1uAu2/4AKE.pdb exists.
Skipping download
```

```
# We assign another random variable that we named "s1.chainA" for the trimmed file
s.chainA <- trim.pdb(structure, chain="A", elety="CA")

# We assign another random variable "s1.b" for extracting the B-factors
s.b <- s.chainA$atom$b

# Here we create a plot of the B-factors
plotb3(s.b, sse=s.chainA, typ="l", ylab="Bfactor")
```



Write a function that can read any pdb files, trim them, extract the specific B-factors and plot the B-factors. When we call the function `plot()`, we get a plot of the B-factors from the pdb file we input

```
plot <- function(x){
  x <- "4AKE"
  structure <- read.pdb(x)
  s.chainA <- trim.pdb(structure, chain="A", elety="CA")
  s.b <- s.chainA$atom$b
  plotb3(s.b, sse=s.chainA, typ="l", ylab="Bfactor")}
plot()
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/1y/t8s3xgz97rg_fclq5rv4c1tw0000gn/T//RtmpQ1uAu2/4AKE.pdb exists.
Skipping download
```