HW 6: R Functions

AUTHOR
Alexis Galano (PID: A17628362)

PUBLISHED
January 25, 2024

R Functions

```
# Can you improve this analysis code?
library(bio3d)
s1 <- read.pdb("4AKE") # kinase with drug
```

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

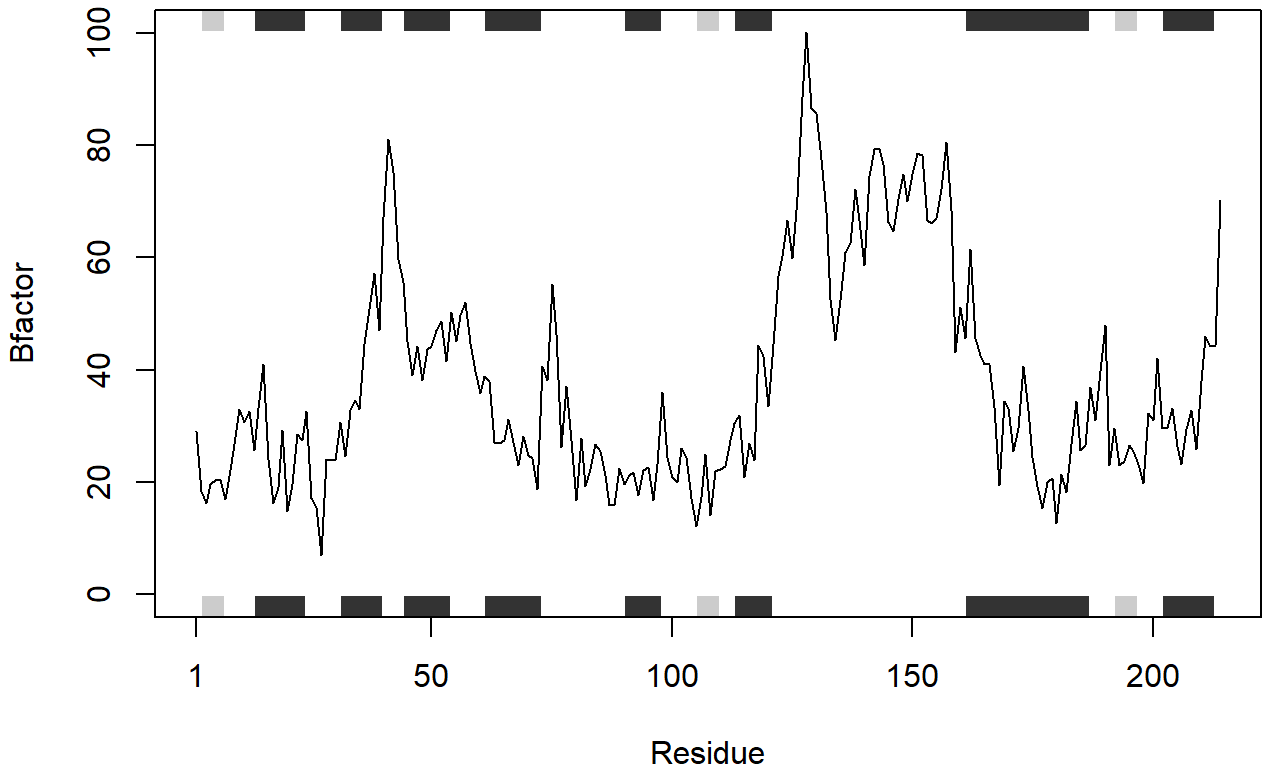
```
s2 <- read.pdb("1AKE") # kinase no drug
```

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

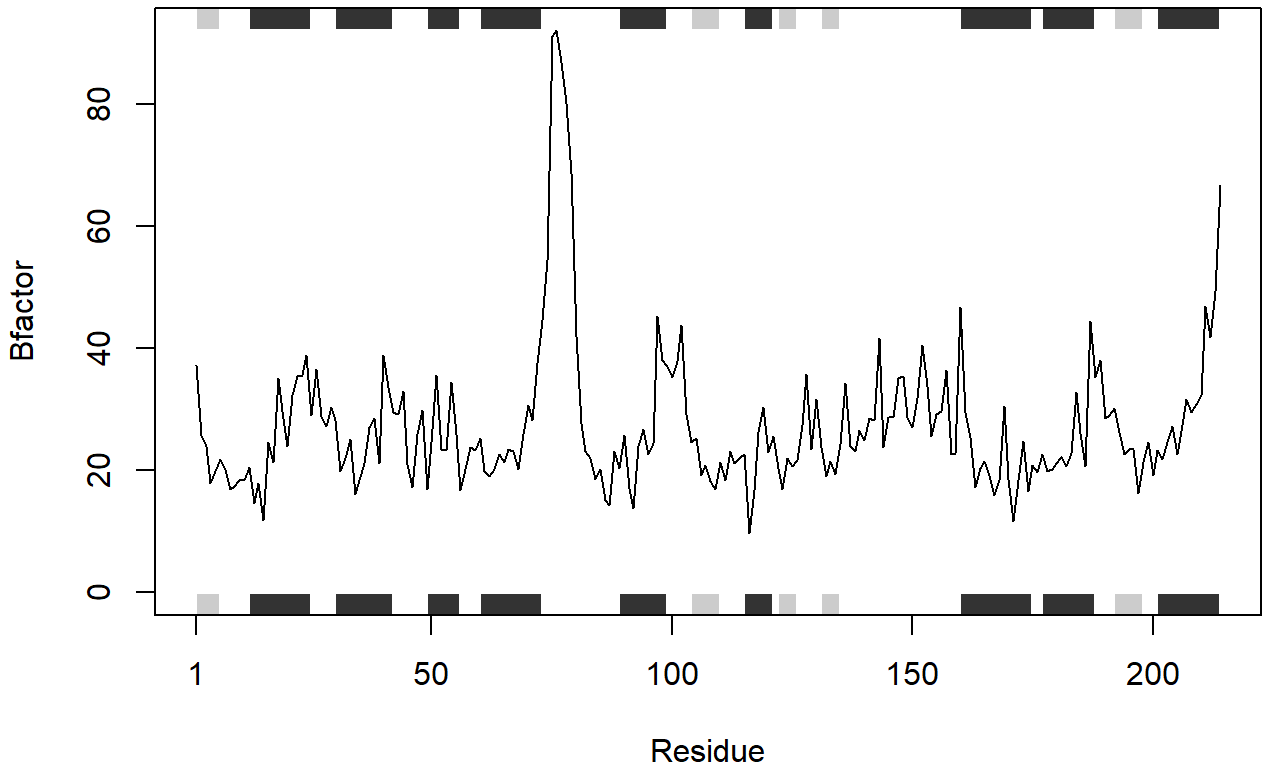
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

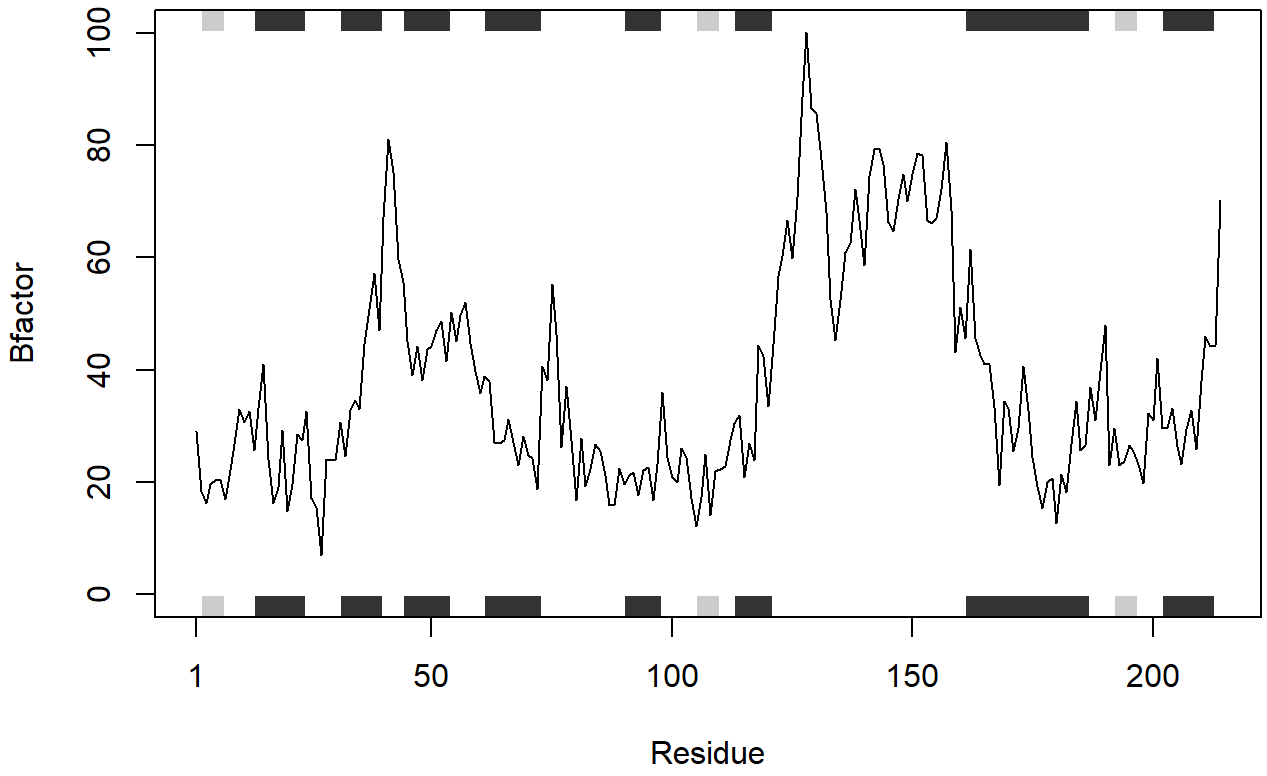
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")

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")
```



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

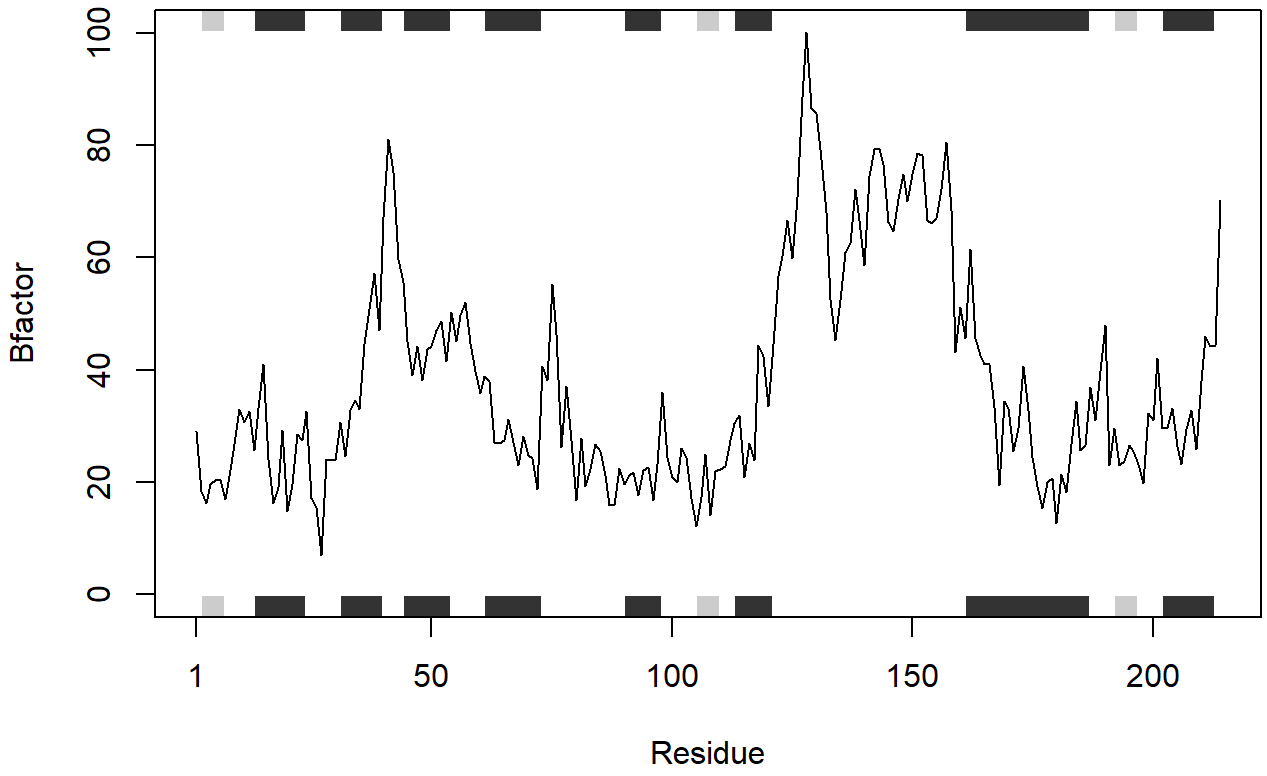
- read.pdb() opens a file with a data bank of proteins

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

- trim.pdb() makes a new smaller PDB object with a subset of atoms from a given larger 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?

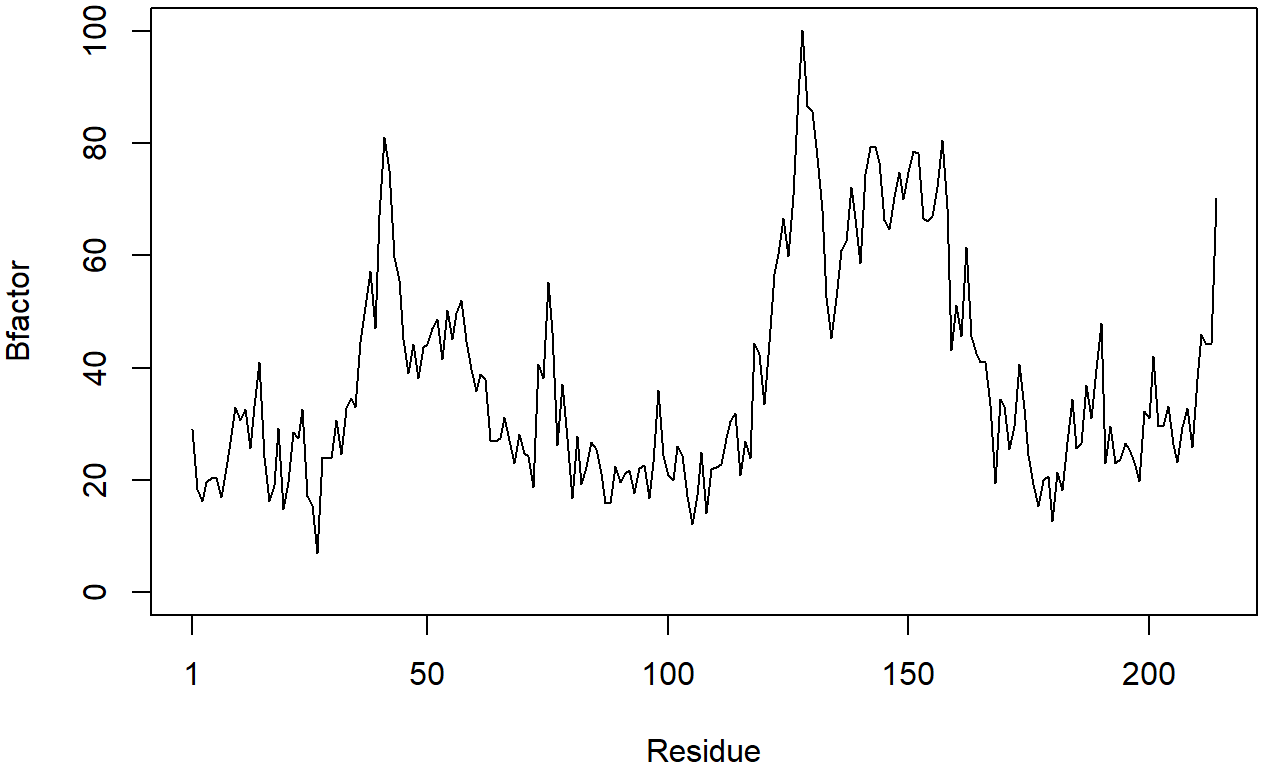
```
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



- We can turn off the marginal black and grey rectangles in the plots by setting sse=NULL instead of sse=s1.chainA

Let's try that below to make sure

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

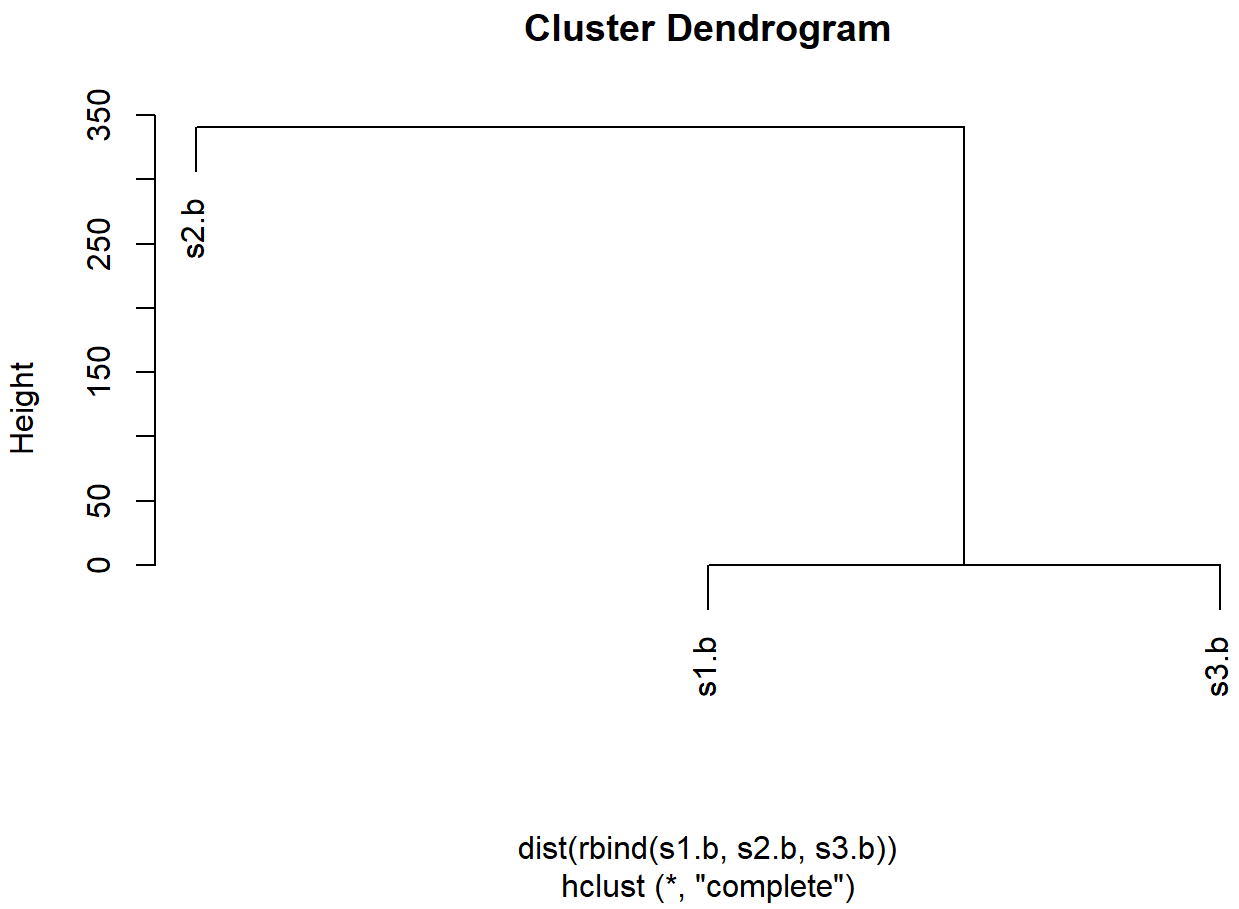


- Yes, we were able to remove the black and white margins by calling sse=NULL. These marginal rectangles may show the location of secondary structure elements.

Q4. What would be a better plot to compare across the different proteins? - We could use RMSD to compare two or more different proteins and their structures.

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

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



- Based on the dendrogram plot, proteins s1.b and s3.b are more similar to each other than to s2.b. This similarity may be due to s1 and s3 being kinases with drugs whereas s2 kinase goes without a drug.

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

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\argal\AppData\Local\Temp\RtmpUBgF7f/4AKE.pdb exists. Skipping download
```

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

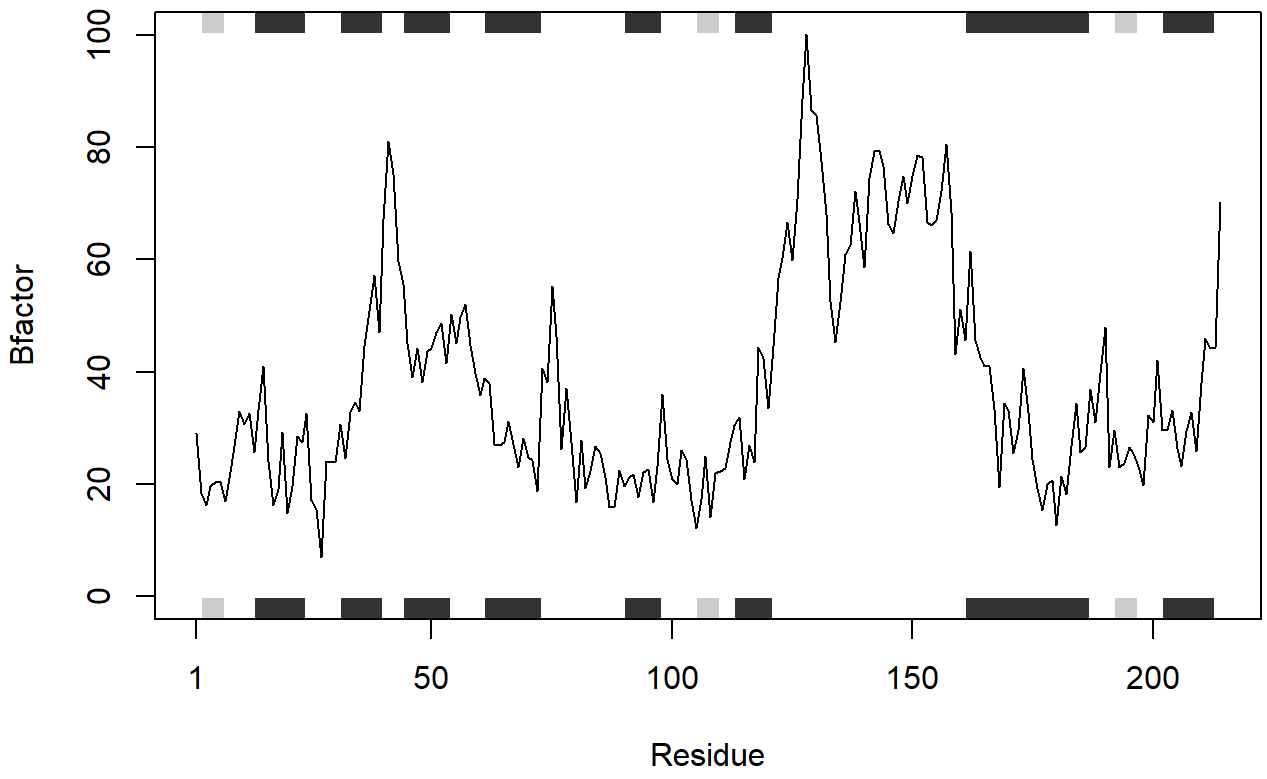
```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\argal\AppData\Local\Temp\RtmpUBgF7f/1AKE.pdb exists. Skipping download

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

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

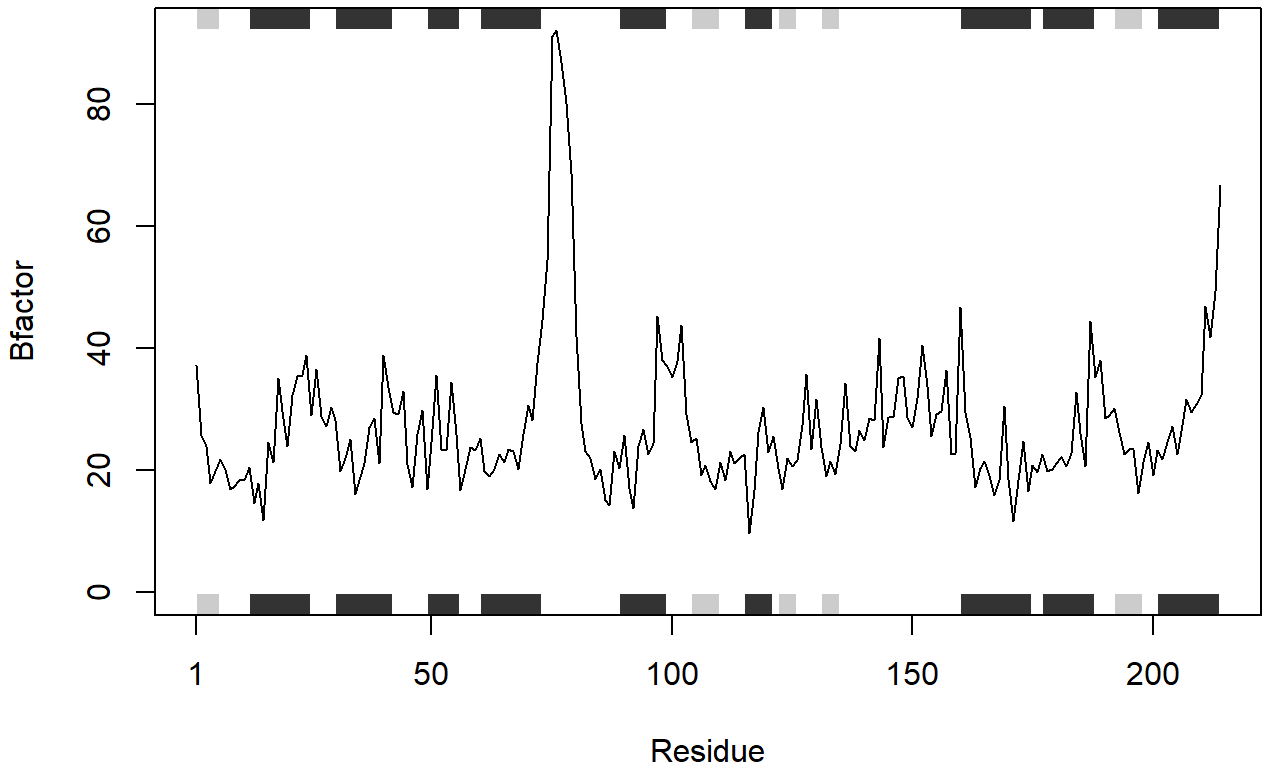
```
  Note: Accessing on-line PDB file
```

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\argal\AppData\Local\Temp\RtmpUBgF7f/1E4Y.pdb exists. Skipping download
```

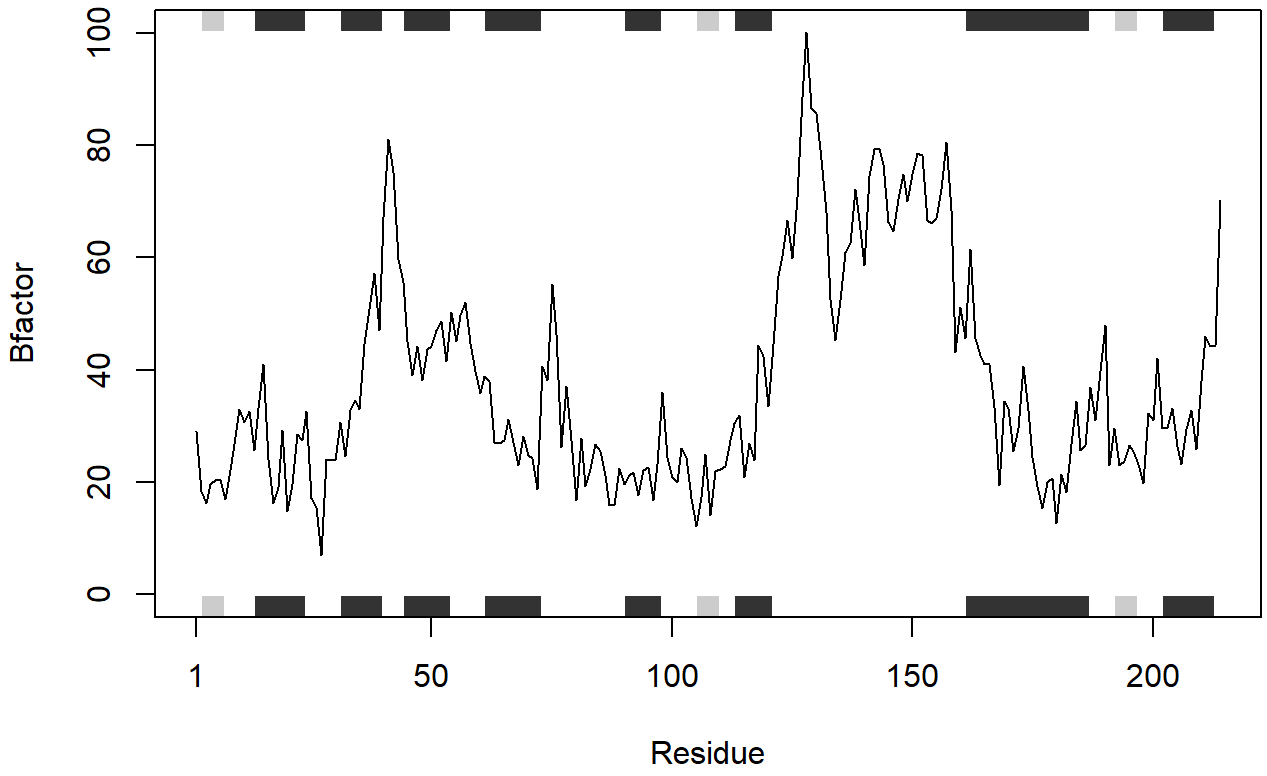
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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




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



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

```
# Original code

library(bio3d)


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

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\argal\AppData\Local\Temp\RtmpUBgF7f/4AKE.pdb exists. Skipping download
```

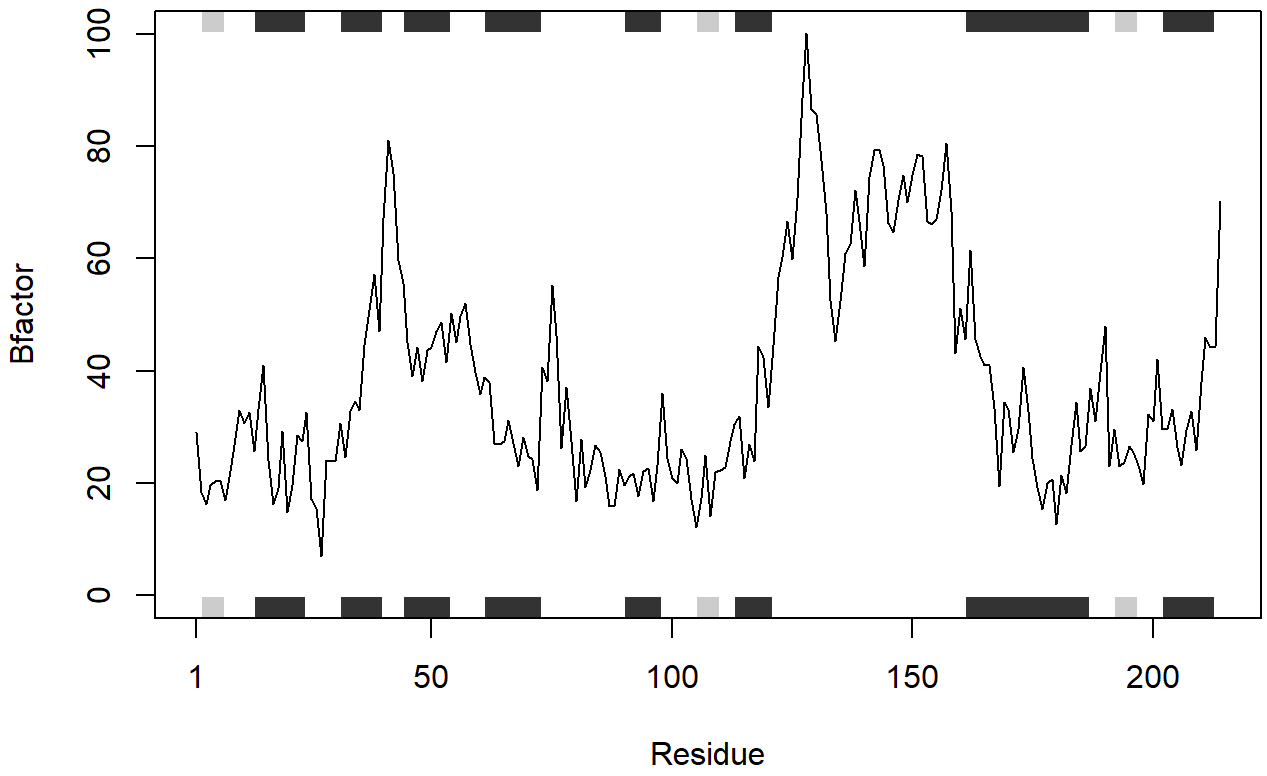
```
s2 <- read.pdb("1AKE") # kinase no drug
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\argal\AppData\Local\Temp\RtmpUBgF7f/1AKE.pdb exists. Skipping download

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

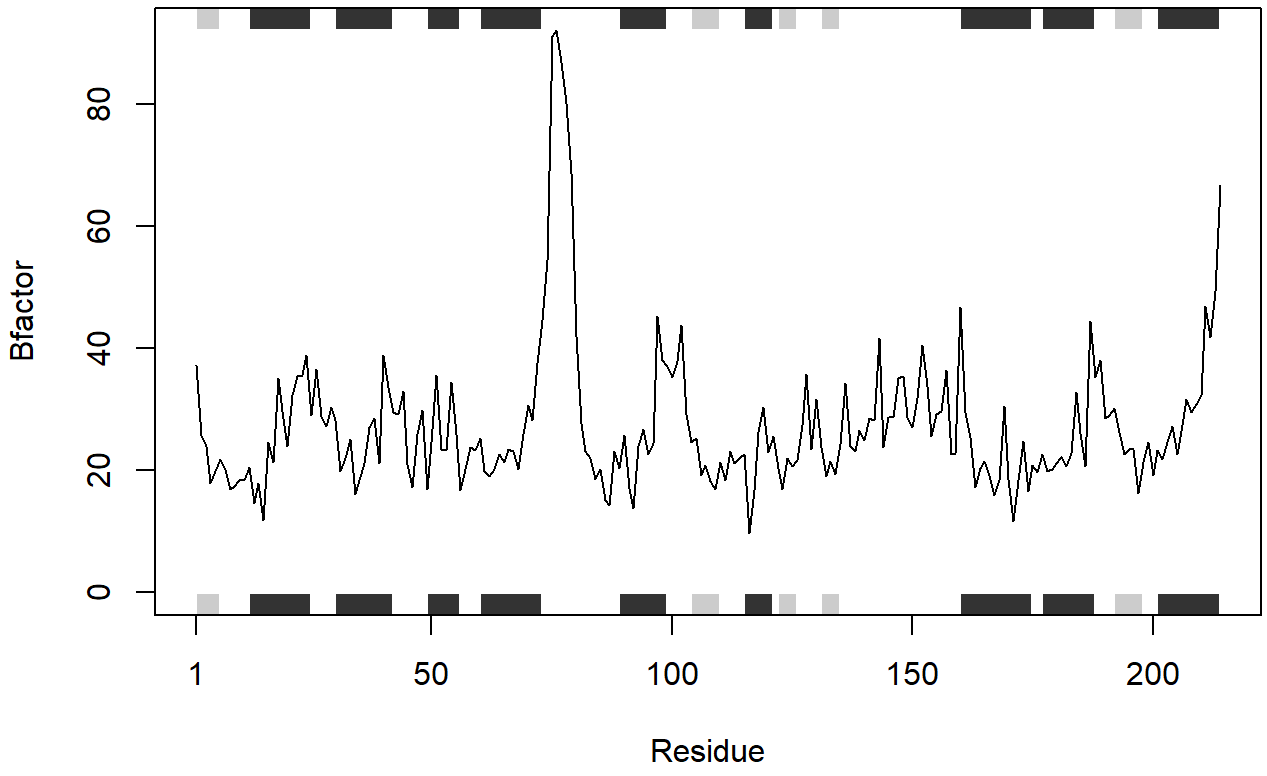
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\argal\AppData\Local\Temp\RtmpUBgF7f/1E4Y.pdb exists. Skipping download
```

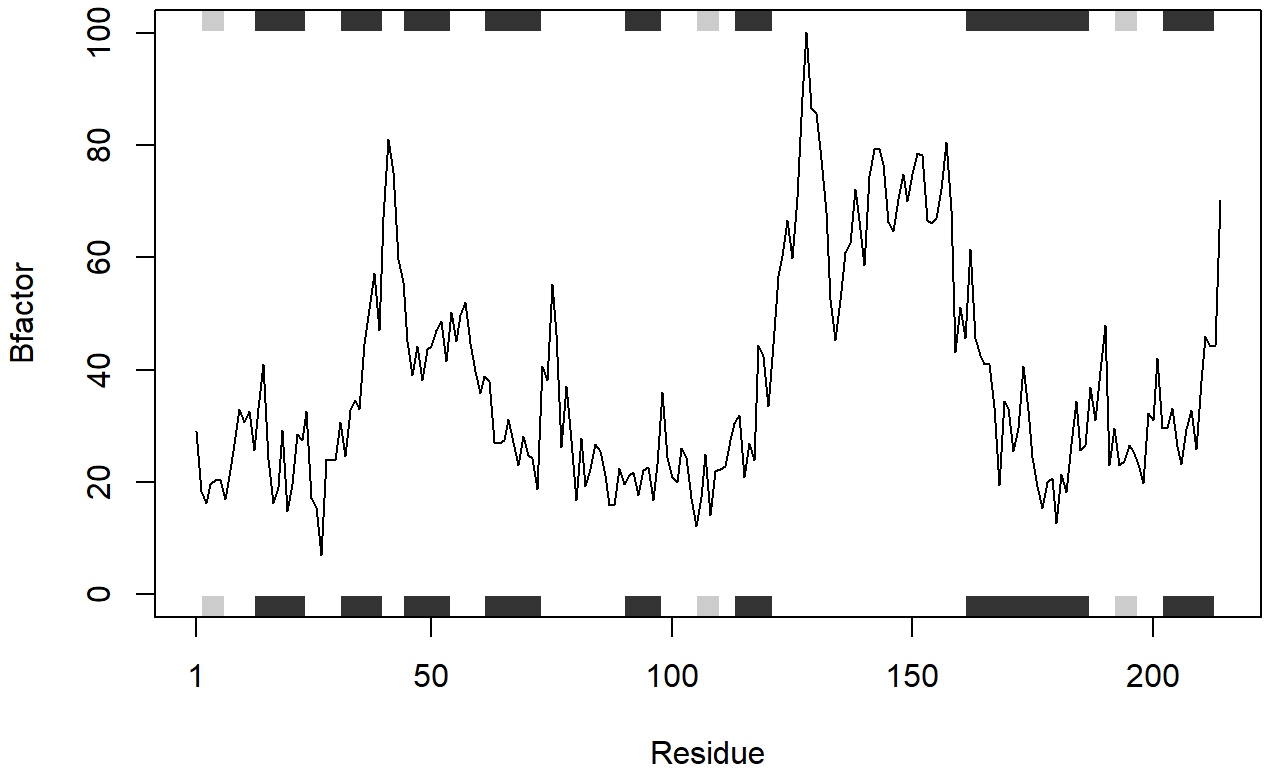
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")

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")
```



We are going to call our function “protein_analyzer” for its ability to assemble an analysis of a set of three desired protein inputs: prot1, prot2, prot3.

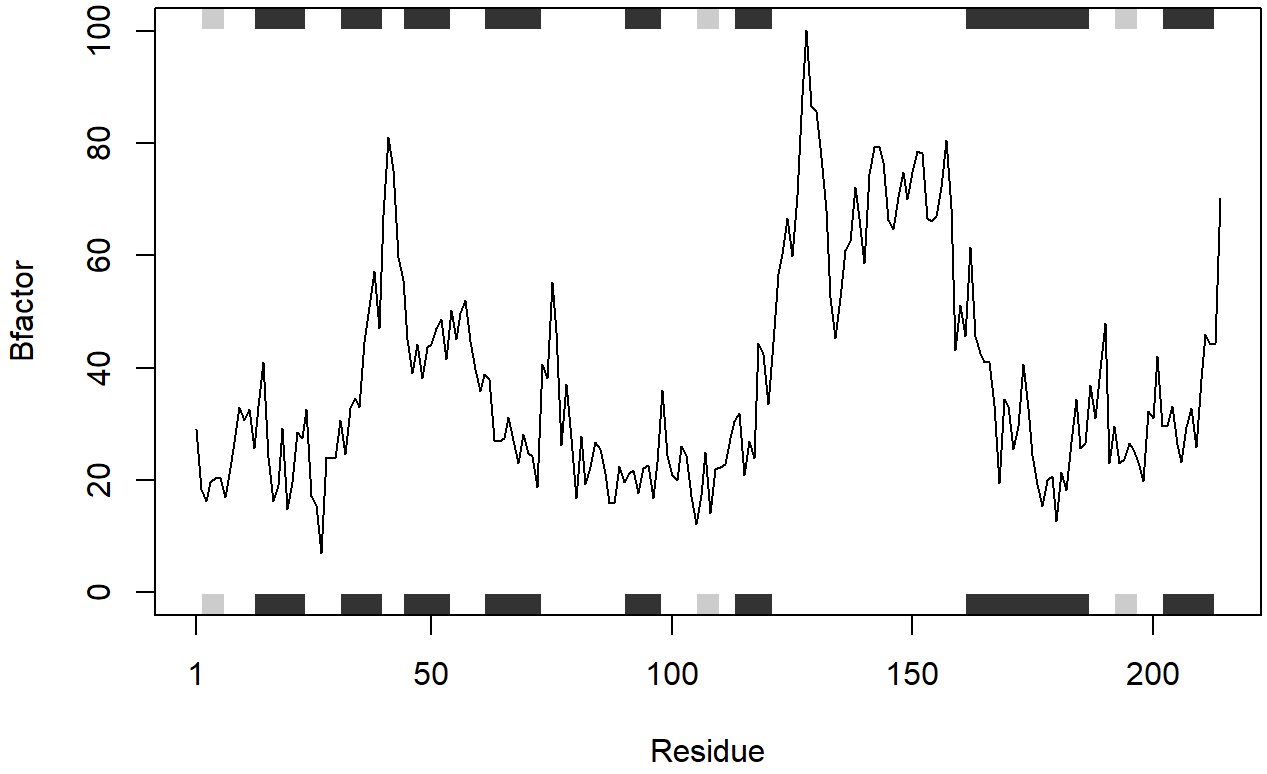
```
protein_analyzer <- function(prot1, prot2, prot3) {

struct1 <- read.pdb(prot1)
struct2 <- read.pdb(prot2)
struct3 <- read.pdb(prot3)

struct1.chainA <- trim.pdb(struct1, chain="A", elety="CA")
struct2.chainA <- trim.pdb(struct2, chain="A", elety="CA")
struct3.chainA <- trim.pdb(struct3, chain="A", elety="CA")

struct1.b <- struct1.chainA$atom$b
struct2.b <- struct2.chainA$atom$b
struct3.b <- struct3.chainA$atom$b

plotb3(struct1.b, sse=struct1.chainA, typ="l", ylab="Bfactor")
plotb3(struct2.b, sse=struct2.chainA, typ="l", ylab="Bfactor")
plotb3(struct3.b, sse=struct3.chainA, typ="l", ylab="Bfactor")
```

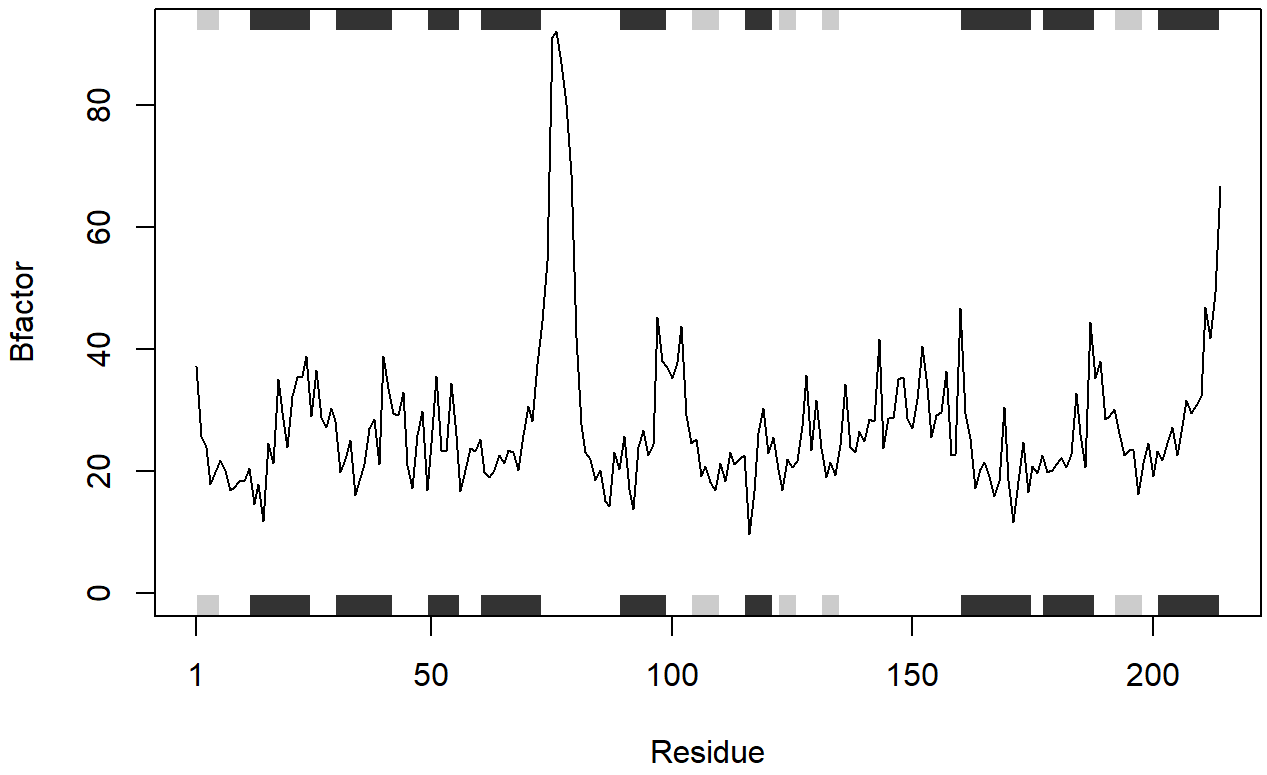
```
}
```

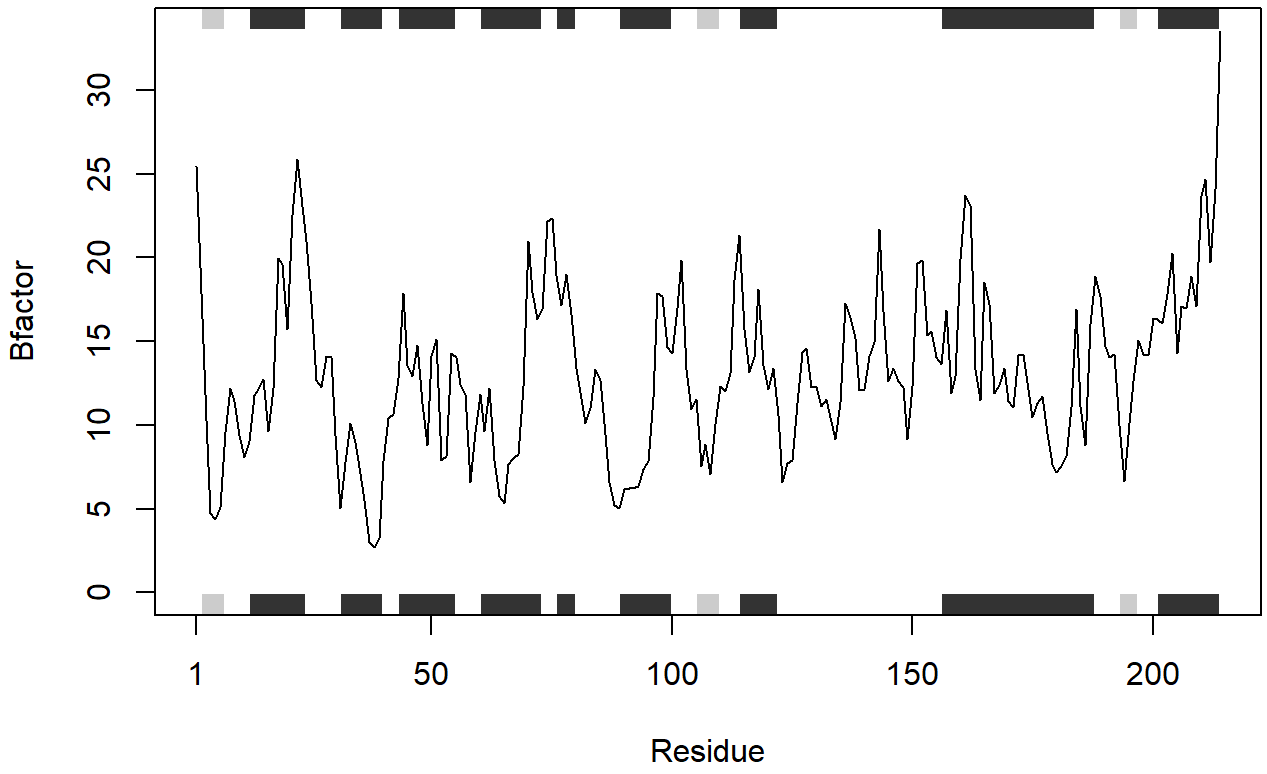
We can now use the function that we made to give us a general analysis of each of our input proteins. Using this function, protein_analyzer, is an easier and faster way to produce our desired output data without having to repeat the body above every single time.

```
protein_analyzer("4AKE", "1AKE", "1E4Y")
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\argal\AppData\Local\Temp\RtmpUBgF7f/4AKE.pdb exists. Skipping download

  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\argal\AppData\Local\Temp\RtmpUBgF7f/1AKE.pdb exists. Skipping download

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

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\argal\AppData\Local\Temp\RtmpUBgF7f/1E4Y.pdb exists. Skipping download
```


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


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



As you can see, our function properly executes and our data and plots are displayed above for each protein we inputted. (output)