Week 5 R Functions

Yash Garodia

06/02/2022

##Section 1

> A. Improve this regular R code by abstracting the main activities in your own new function. Note, we will go through this example together in the formal lecture. The main steps should entail running through the code to see if it works, simplifying to a core working code snippet, reducing any calculation duplication, and finally transferring your new streamlined code into a more useful function for you.

```
# (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))
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
```

```
#First I created a function that I can apply to each row
mod <- function(x){
  (x - min(x))/(max(x)-min(x))
}
```

```
# Then I applied the function to each row
a <- mod(df$a)
b <- mod(df$b)
c <- mod(df$c)
d <- mod(df$d)
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
```

The results I got are the same as the results in the first code, so my answer is correct.

> B. Next improve the below example code for the analysis of protein drug interactions by abstracting the main activities in your own new function. Then answer questions 1 to 6 below. It is recommended that you start a new Project in RStudio in a new directory and then install the bio3d package noted in the R code below (N.B. you can use the command install.packages("bio3d") or the RStudio interface to do this). Then run through the code to see if it works, fix any copy/paste errors before simplifying to a core working code snippet, reducing any calculation duplication, and finally transferring it into a more useful function for you.

```
# Can you improve this analysis code?
library(bio3d)
```

```
## Warning: package 'bio3d' was built under R version 4.0.2
```

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

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

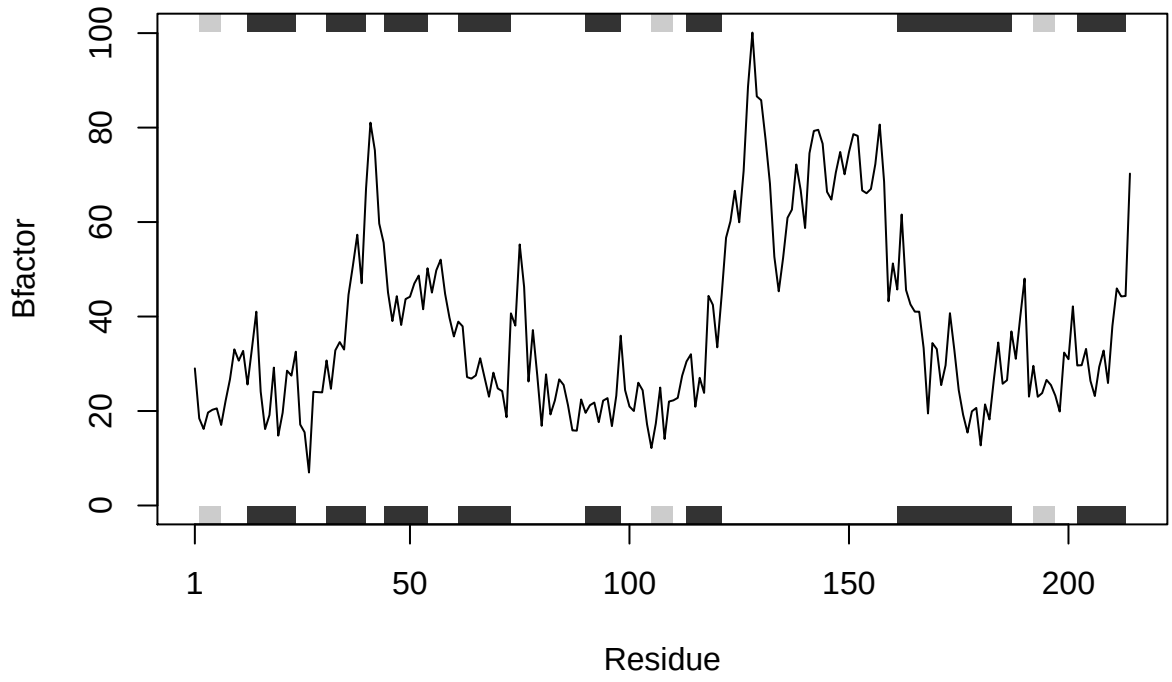
```
s2 <- read.pdb("1AKE") # kinase no drug
```

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

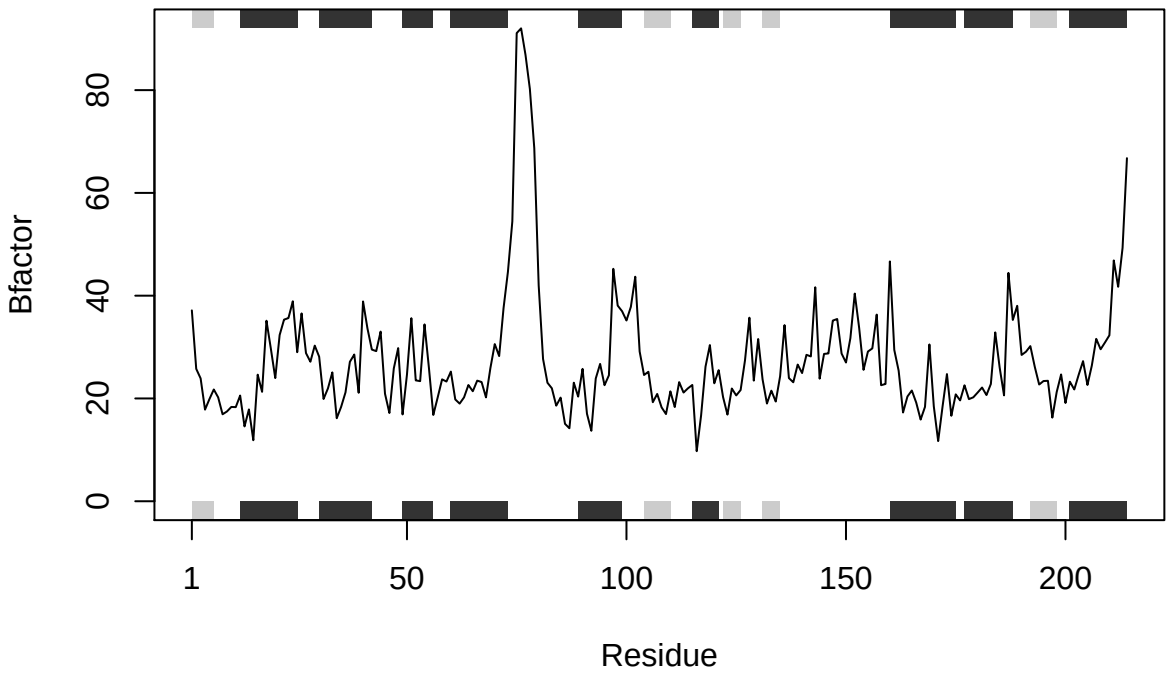
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

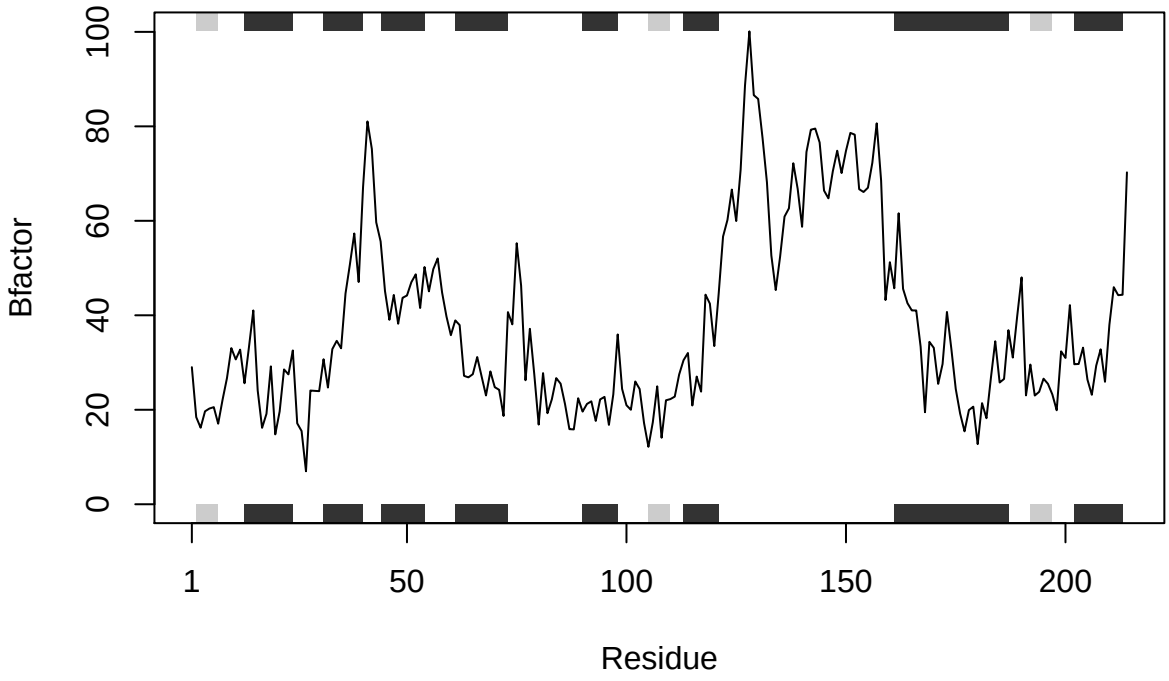
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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




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



In order to create a universal function, I just replaced s1,s2 and s3 with x. Lets see how this works:

```
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/hy/
## dr_5sffs0c5dw0r707pdz9kr0000gn/T//RtmpjDXfMd/4AKE.pdb exists. Skipping download
```

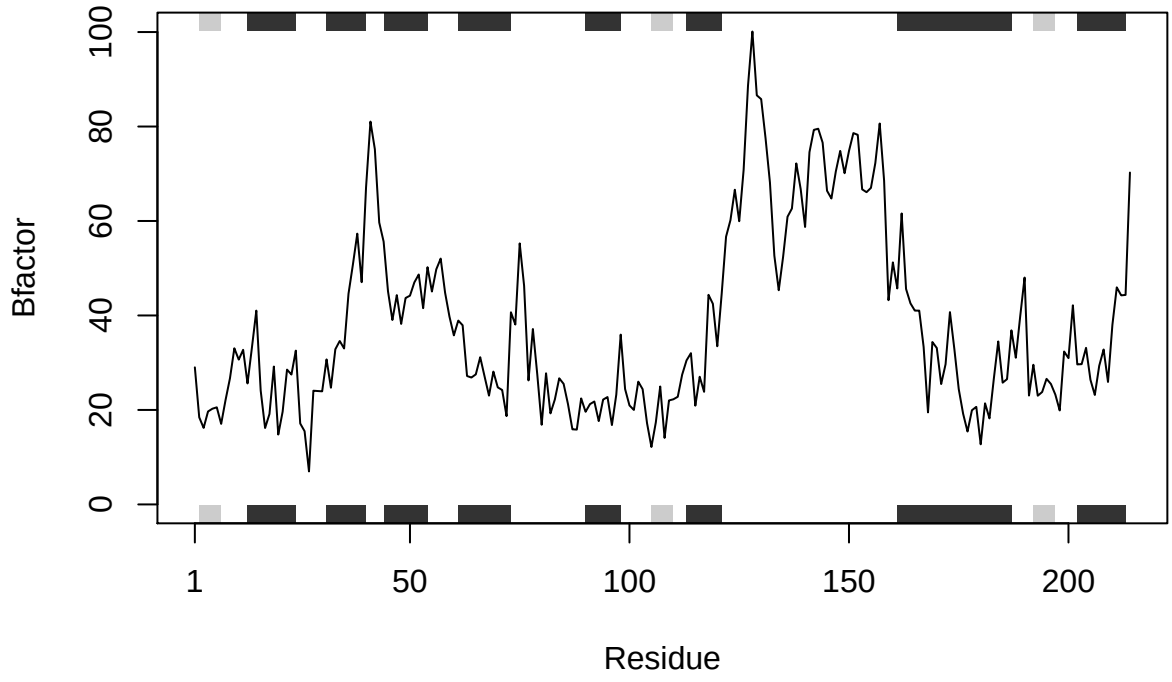
```
s2 <- read.pdb("1AKE") # kinase no drug
```

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/hy/
## dr_5sffs0c5dw0r707pdz9kr0000gn/T//RtmpjDXfMd/1AKE.pdb exists. Skipping download

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

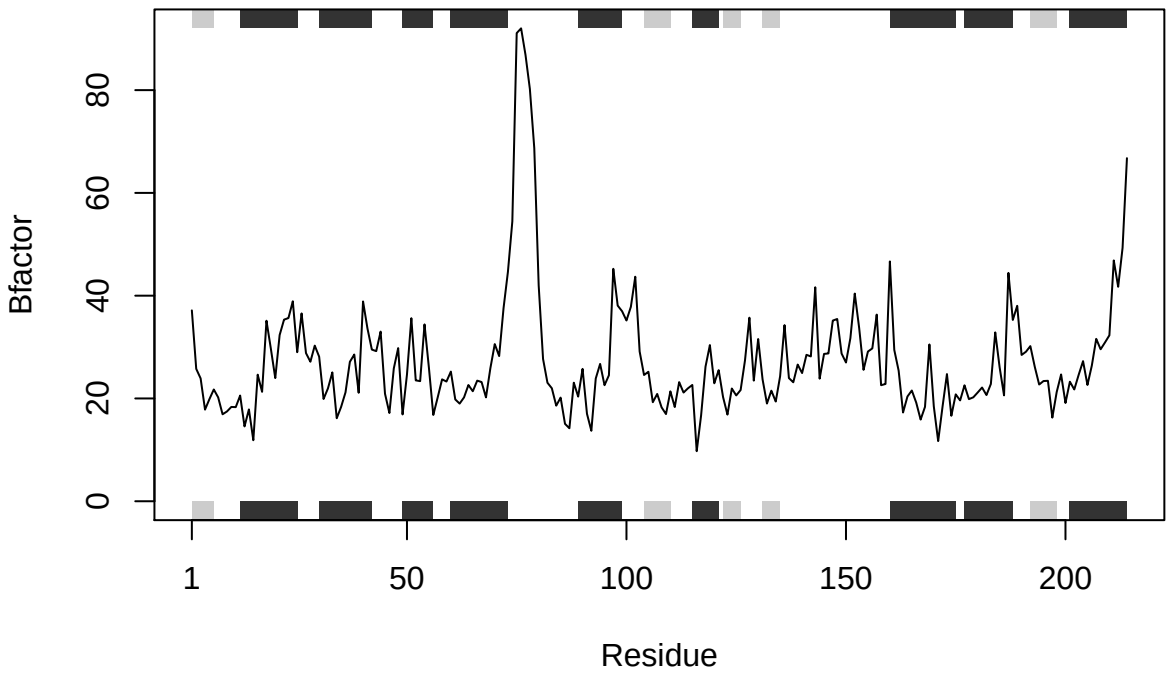
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/hy/
## dr_5sffs0c5dw0r707pdz9kr0000gn/T//RtmpjDXfMd/1E4Y.pdb exists. Skipping download
```

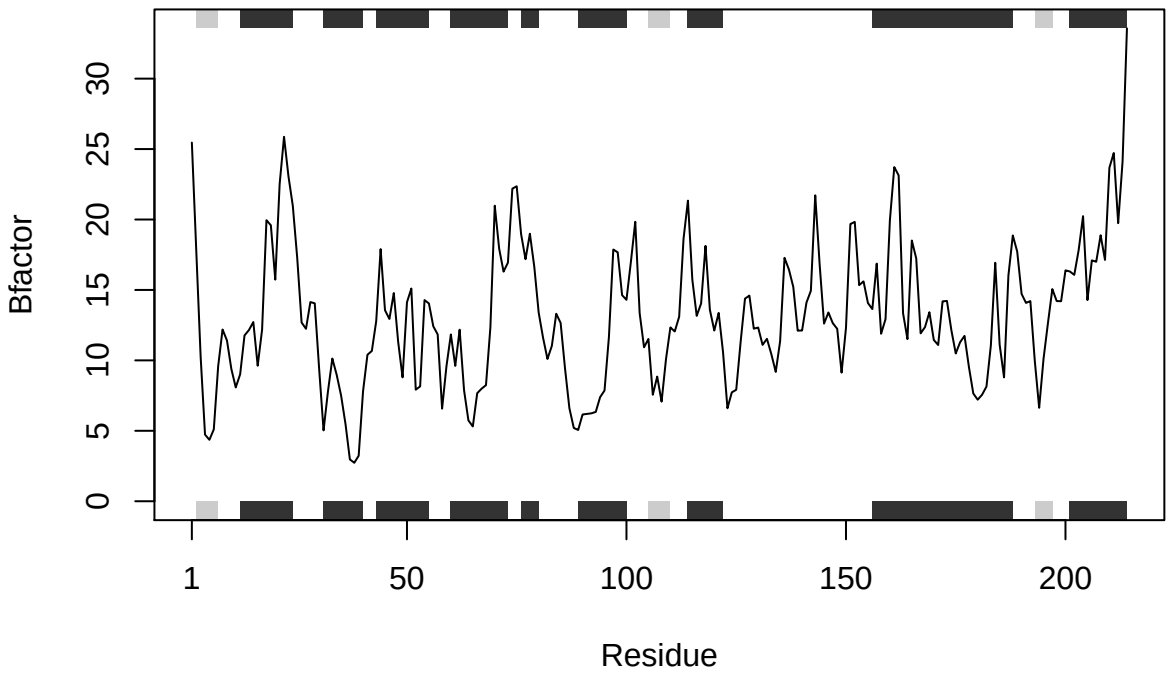
```
prot_analysis <- function(x){
  x.chainA <- trim.pdb(x, chain="A", elety="CA")
  x.b <- x.chainA$atom$b
  plotb3(x.b, sse = x.chainA, typ="l", ylab="Bfactor")
}
prot_analysis(s1)
```




```
prot_analysis(s2)
```




```
prot_analysis(s3)
```



Answering the questions given: **Q1. What type of object is returned from the read.pdb() function?**

Ans. It returns a list of class "pdb: containing the ATOM and HETATM data, start, end and lengths of H and E type SSE, sequence from the SEQRES field, and additional remarks.

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

Ans. It produces a new smaller PDB object, containing a subset of atoms, from a 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?

Ans. They represent secondary structural element annotations of proteins. Remove the sse parameter to turn off the rectangles.

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

Ans. The plot without rectangles as the rectangles would most likely overlap and would be confusing to understand.

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

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

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

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/hy/
## dr_5sffs0c5dw0r707pdz9kr0000gn/T//RtmpjDXfMd/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/hy/
## dr_5sffs0c5dw0r707pdz9kr0000gn/T//RtmpjDXfMd/1AKE.pdb exists. Skipping download
```

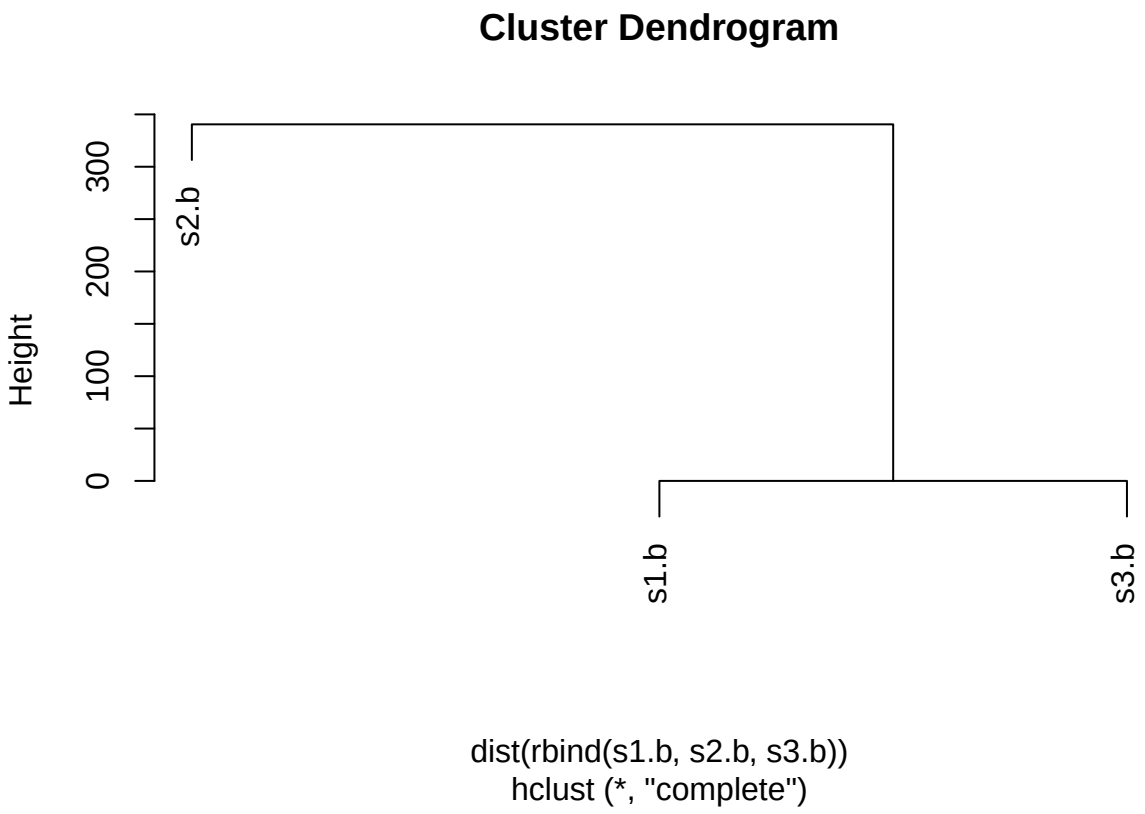
```
##    PDB has ALT records, taking A only, rm.alt=TRUE
```

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

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

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/hy/
## dr_5sffs0c5dw0r707pdz9kr0000gn/T//RtmpjDXfMd/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
hc <- hclust( dist( rbind(s1.b, s2.b, s3.b) ) )
plot(hc)
```



> Ans. s1 and s3 are more similar to each other because the distance between them in the dendrogram is lower.

Q6. How would you generalize the original code above to work with any set of input protein structures? 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. (See class lecture 9 for further details)

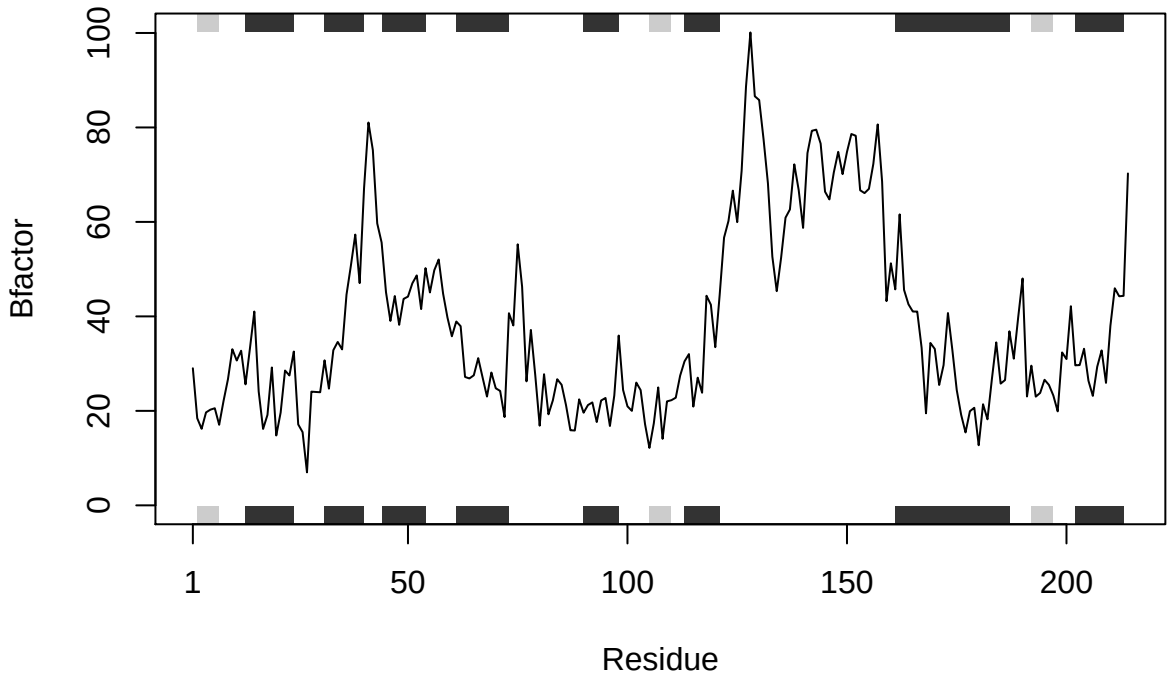
Ans:

```
library(bio3d)
#Inputs to this function are things that the computer cannot already know. In this case, it would be th

univ_prot_analysis <- function(prot, chain, elety){
#First the function reads the PDB ID to identify the PDB structure object, and stores the details in a
  x <- read.pdb(prot)
# It then trims the structure object, only leaving the desired chain and its respective atoms.
  x.chain <- trim.pdb(x, chain = chain, elety = elety)
# It then further restricts the protein structure to the B factor in that chain.
  x.b <- x.chain$atom$b
#Finally the function creates a line plot that maps the Bfactor
  plotb3(x.b, sse = x.chain, typ="l", ylab="Bfactor")
}
#To ensure that my answer was correct, I compared this function of mine to the earlier function I made
univ_prot_analysis("4AKE", "A", "CA")
```

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

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/hy/
## dr_5sffs0c5dw0r707pdz9kr0000gn/T//RtmpjDXfMd/4AKE.pdb exists. Skipping download
```

```
prot_analysis(s1)
```




```
#The answer was the same so I have successfully created a concise, effective code!
```