Week3HW6Question6

Sarah Tareen

2023-04-25

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

Q2 What does the trim.pdb() function do? The trim.pdb() function creates a new PDB object based on the selection of atoms given in the arguments.

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 input parameter sse would turn off the marginal black and grey rectangles. In this case, it represents which residues have chain A.

Q4 What would be a better plot to compare across the different proteins? A better plot could be a scatterplot of the different proteins that has smooth lines so its easier to compare them side by side.

Q5 Which proteins are more similar to each other in their B-factor trends. How could you quantify this? HINT: try the rbind(), dist() and hclust() functions together with a resulting dendrogram plot. Look up the documentation to see what each of these functions does. - rbind() combines data structured by columns or rows. - dist() find the distances between the rows of a data matrix - hclust() makes a dendrogram plot of the dissimilarities

```
## We can use these PDB files for the proteins
library(bio3d)
s1 <- read.pdb("4AKE") # kinase with drug
```

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

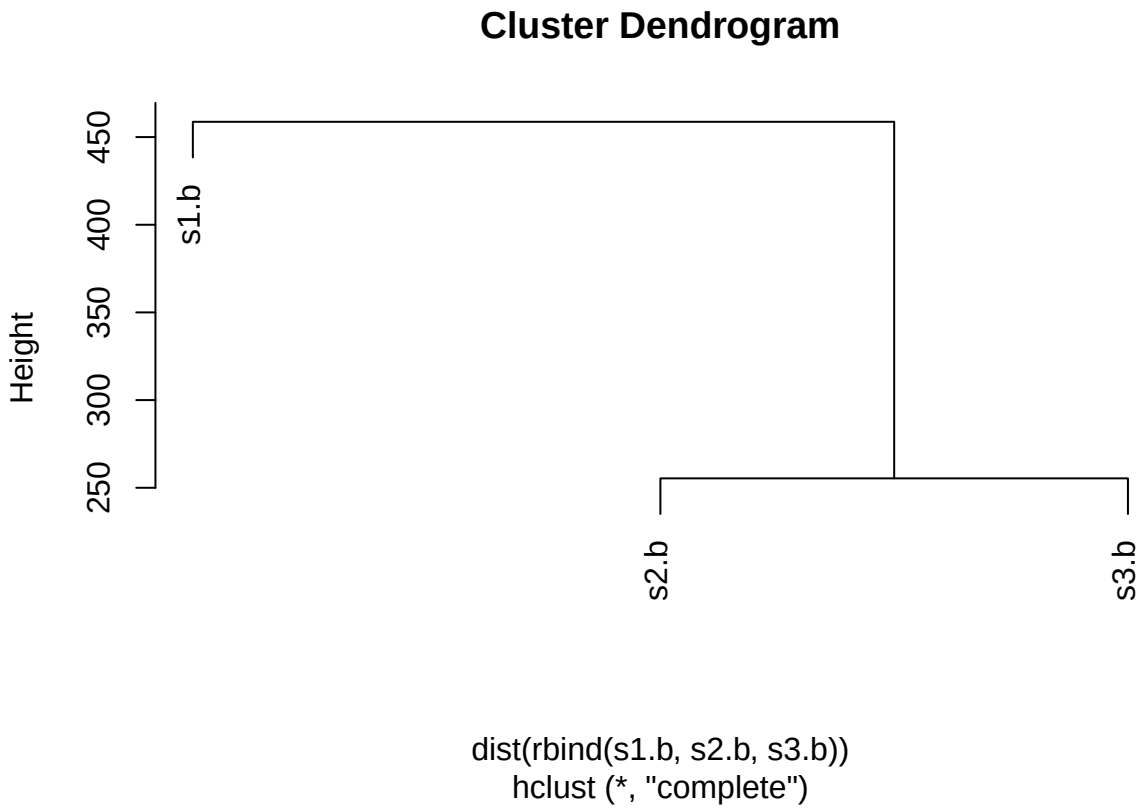
```
s2 <- read.pdb("1AKE") # kinase no drug
```

```
##   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")
## changed the s1 to s3
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

## This is the correlation between the B factors
hc <- hclust( dist( rbind(s1.b, s2.b, s3.b) ) )
plot(hc)
```



Since s2.b and s3.b are more closely related then the proteins "1AKE" and "1E4Y" are more related to each other.

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

This is the original code with the copy paste error fixed:

```
# Can you improve this analysis 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\Owner\AppData\Local\Temp\RtmpCs5gXg/4AKE.pdb exists. Skipping download
```

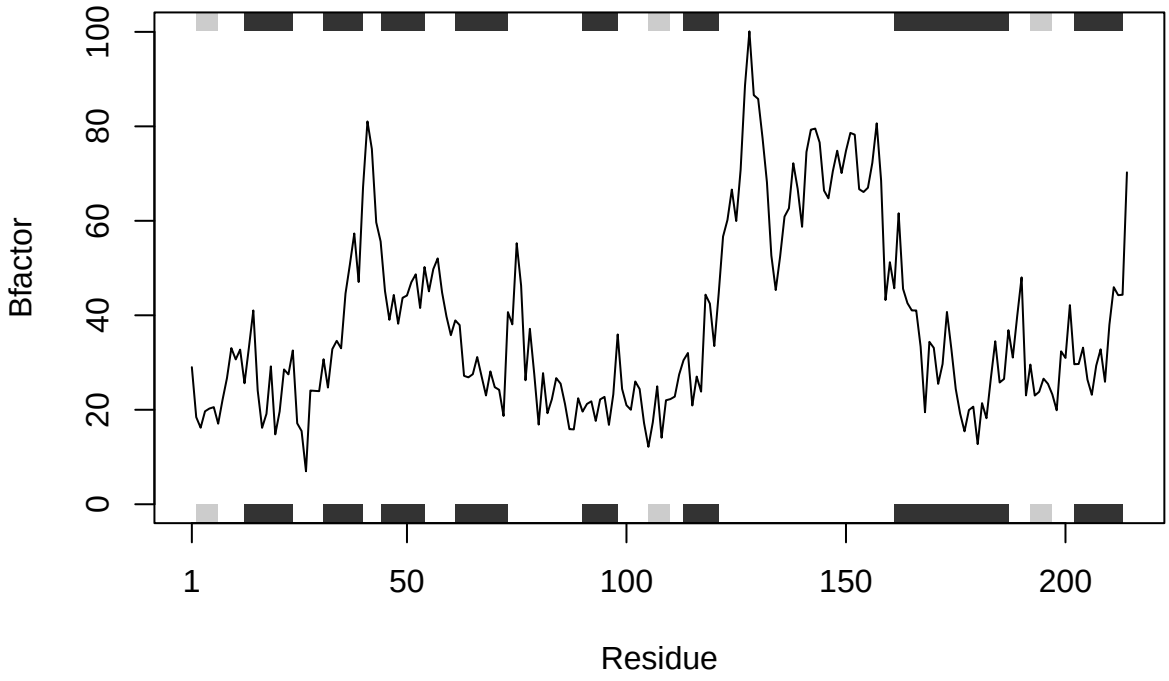
```
s2 <- read.pdb("1AKE") # kinase no drug
```

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
## C:\Users\Owner\AppData\Local\Temp\RtmpCs5gXg/1AKE.pdb exists. Skipping download

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

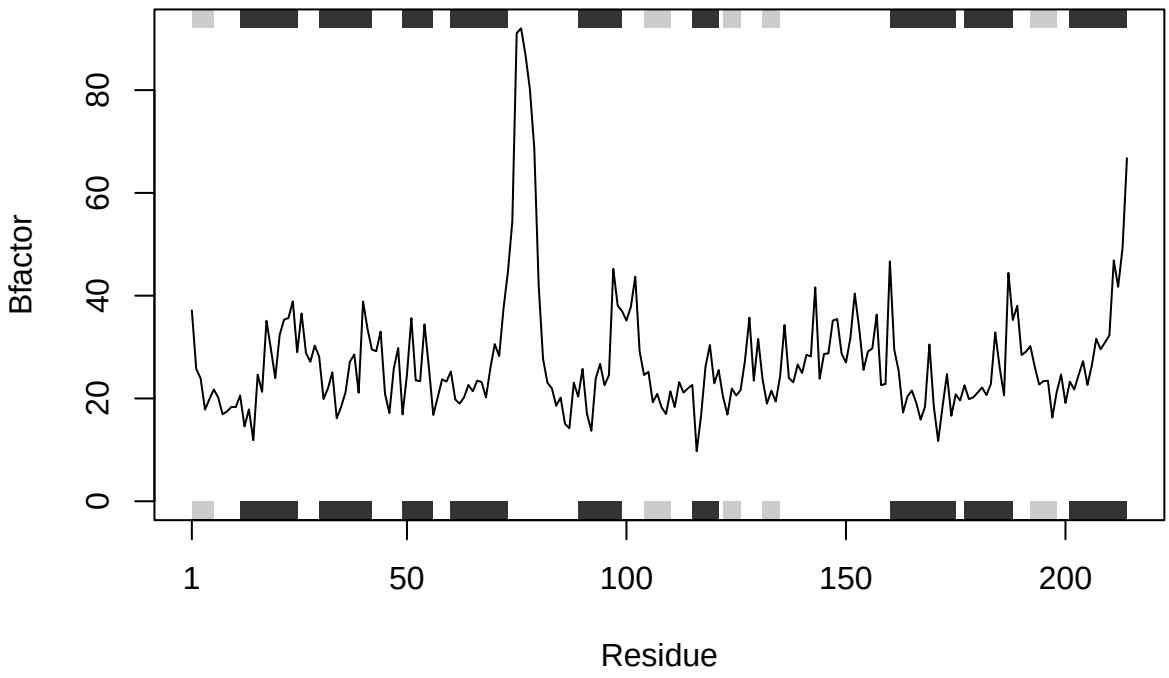
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
## C:\Users\Owner\AppData\Local\Temp\RtmpCs5gXg/1E4Y.pdb exists. Skipping download
```

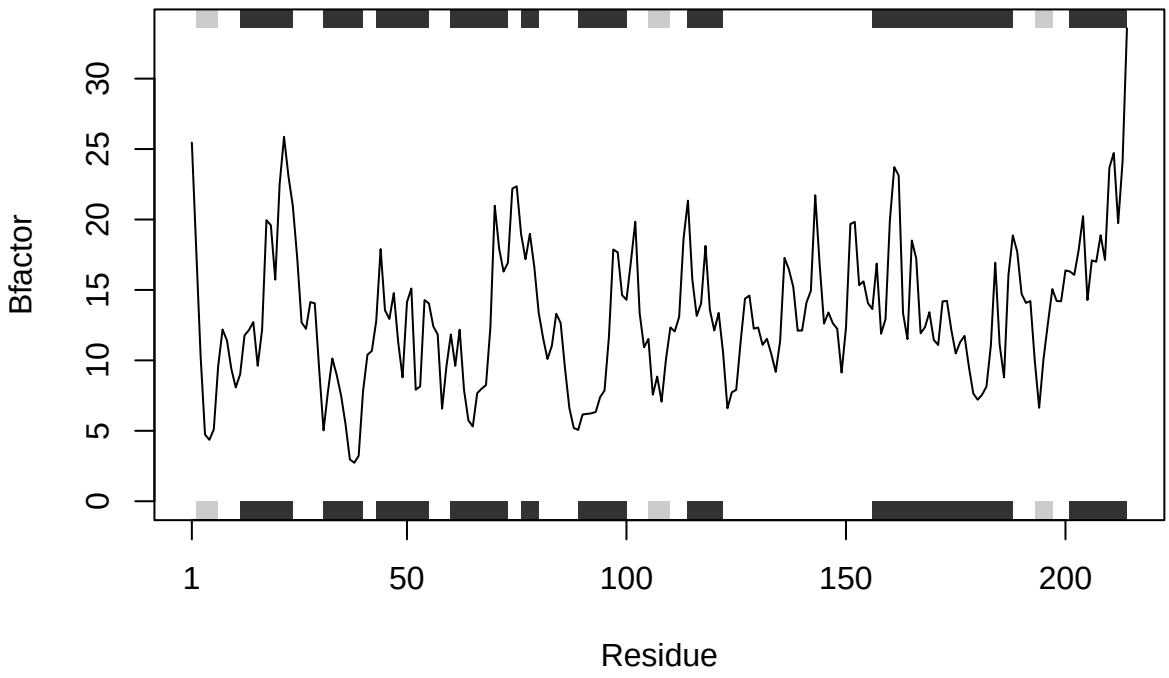
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
## changed the s1 to s3
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")

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")
```

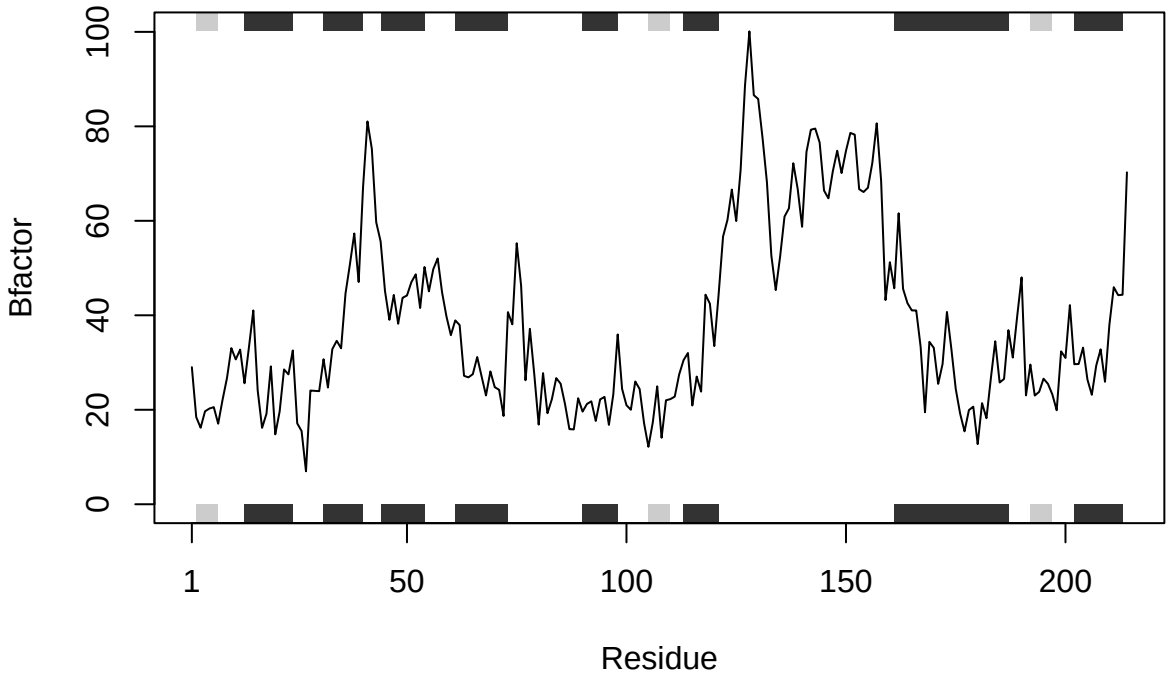


Now we make a working code snippet by generalizing the variables in the code so it can work with any set of protein structures.

```
## We can use this protein as an example to test our code snippet.
file <- "4AKE"

## This is our generalized code
output <- read.pdb(file)
```

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
## C:\Users\Owner\AppData\Local\Temp\RtmpCs5gXg/4AKE.pdb exists. Skipping download
```

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

output.b <- output.chainA$atom$b

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



Since the code snippet works we can turn it into a function:

```
#' Finds the number of B factors in a PDB protein file after trimming the output to select
#' the PDB proteins based on chain A with atoms CA.
#' @param file The PDB file from bio3d
#'
#' @return A plot of the residues of the protein vs the number of
#' B factors also showing the number of the residues with chain A at the top
#' @export
#'
#' @examples
#' file <- "4AKE"
#' plotBfactor(file)
#'
plotBfactor <- function(file) {
  output <- read.pdb(file)

  ## Make new PDB object selecting chain A and atoms CA
  output.chainA <- trim.pdb(output, chain="A", elety="CA")

  ## Only select B atoms from chain A
  output.b <- output.chainA$atom$b

  ## Line plot that shows which residues have chain A at the top
  plotb3(output.b, sse=output.chainA, typ="l", ylab="Bfactor")
}
```