RMarkdownHW06

Patrick Tran

2023-01-30

Protein Drug Interactions

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

First, we take a look at the lines of code that get repeated.

```
s1 <- read.pdb("4AKE")
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```

Simplify to work with a generic vector named "x"

```
x <- read.pdb("file")
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")
```

After we have identified this, we are able to start on turning this into a function.

protein.drug.plot will be the name of our function. This will allow us to visualize specific protein drug interactions utilizing protein PDB data.

The function **protein.drug.plot** input is a 4-character PDB ID or identifier. It analyzes the protein's information coming from the database as long as the input of the function is consistent with the ID shown on PDB. **protein.drug.plot** simplifies visualizing data and reduces code duplication. The output will be a singular plot. Residue is on the x-axis and Bfactor is on the y-axis.

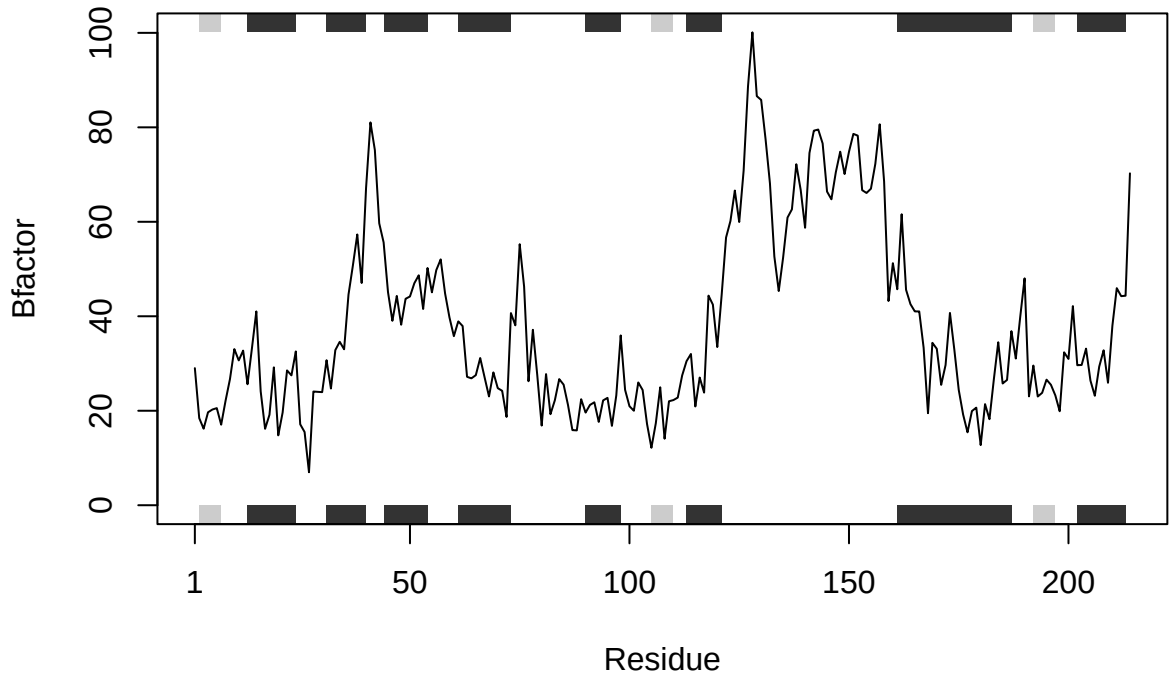
```
#The input for the function is "file"
protein.drug.plot <- function(file) {
 x <- read.pdb(file)
x.chainA <- trim.pdb(x, chain="A", elety="CA")
x.b <- x.chainA$atom$b
plotb3(x.b, sse=x.chainA, typ="l", ylab="Bfactor")
}
```

Test if the function output matches with the original analysis code.

```
#Original analysis code
library(bio3d)
s1 <- read.pdb("4AKE")
```

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

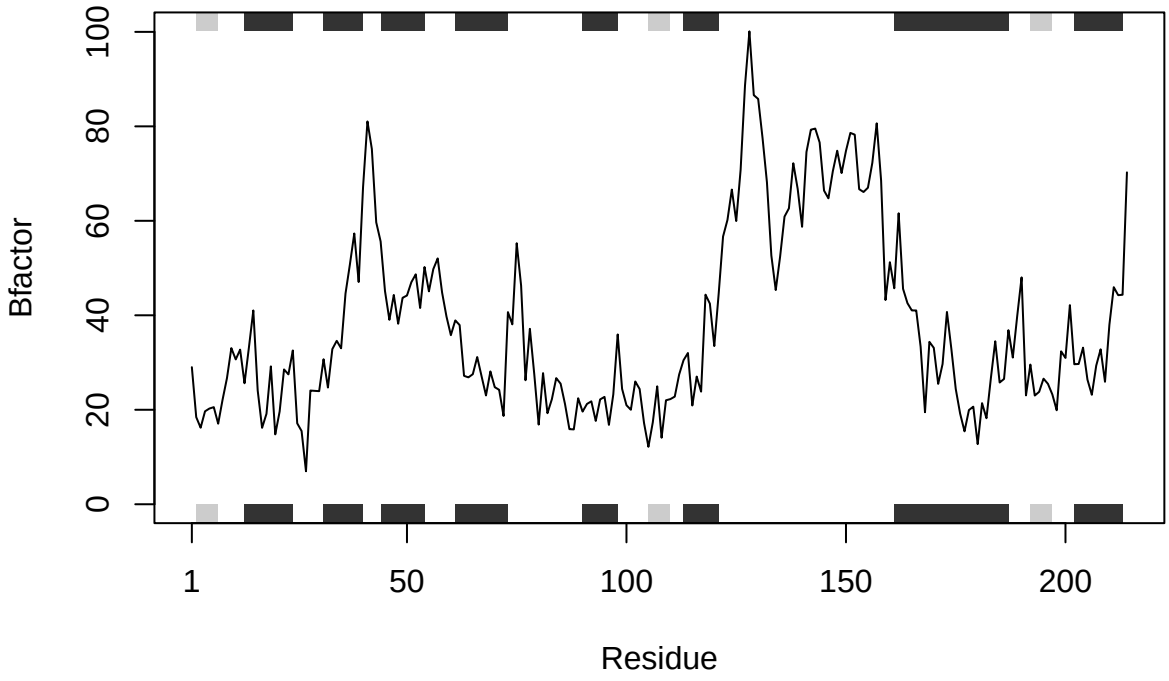
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




```
#Function
protein.drug.plot("4AKE")
```

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
## C:\Users\ptran\AppData\Local\Temp\Rtmpek8eFQ/4AKE.pdb exists. Skipping download
```

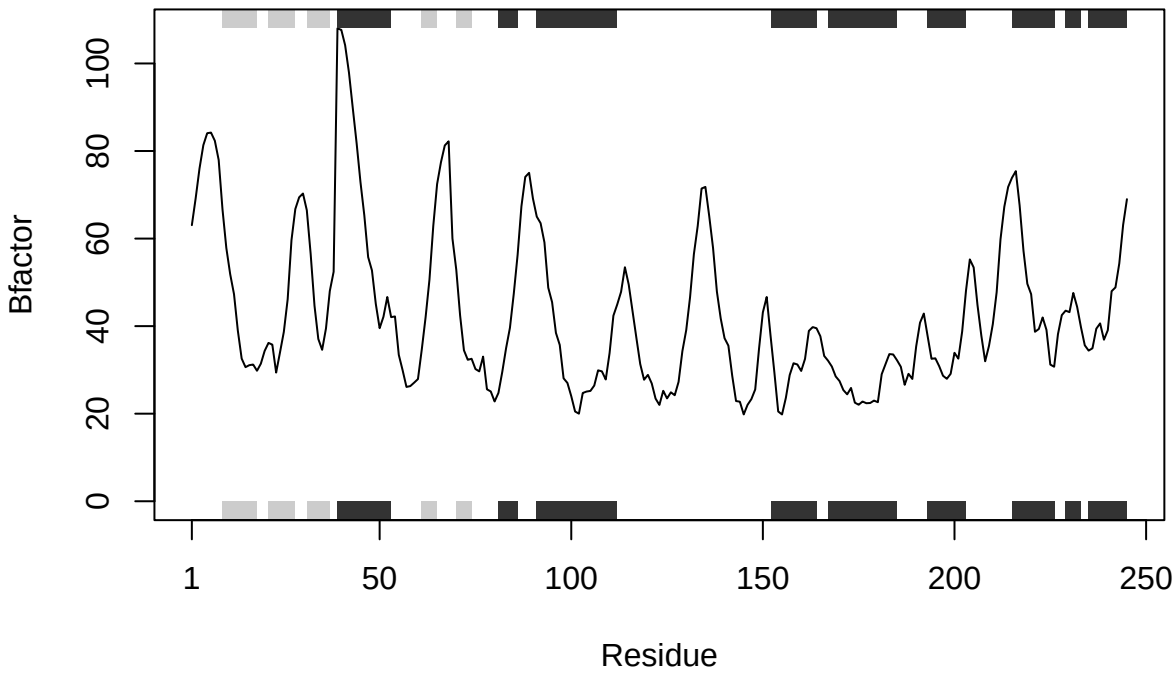


The outputs are identical to each other. We can conclude that the function is behaving as desired.

We can also input any protein PDB data using the ID. As an example we will be using “3DTC”

```
protein.drug.plot("3DTC")
```

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


Bfactor
0
20
40
60
80
100
1
50
100
150
200
250
Residue