LE6_HW

RUNQI ZHANG

Table of contents

Q6. How would you generalize the original code above to work with any set of input **1**
Student's answer 6
test run 7

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

protein structures?

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

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

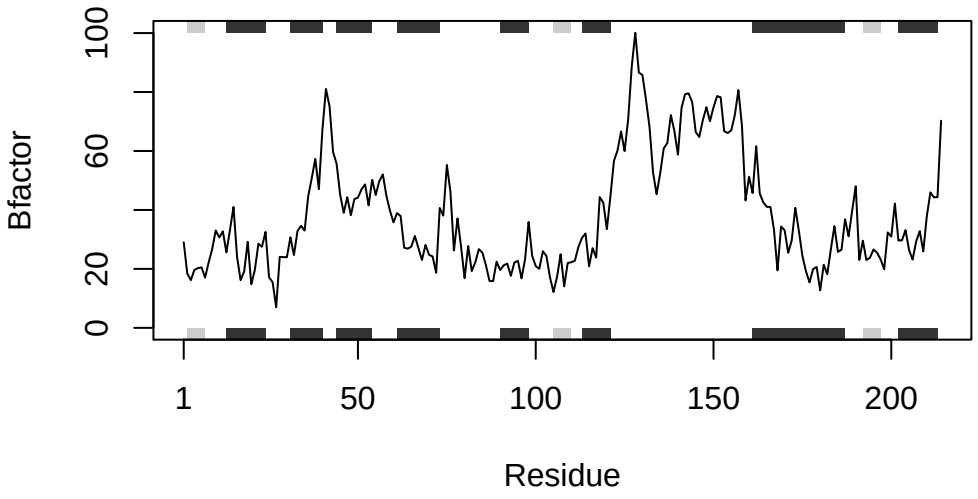
```
s2 <- read.pdb("1AKE") # kinase no drug
```

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

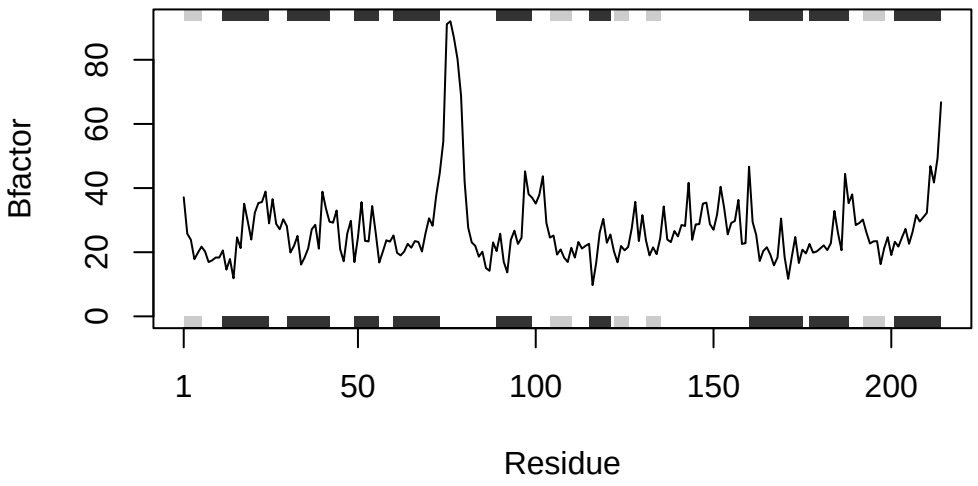
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

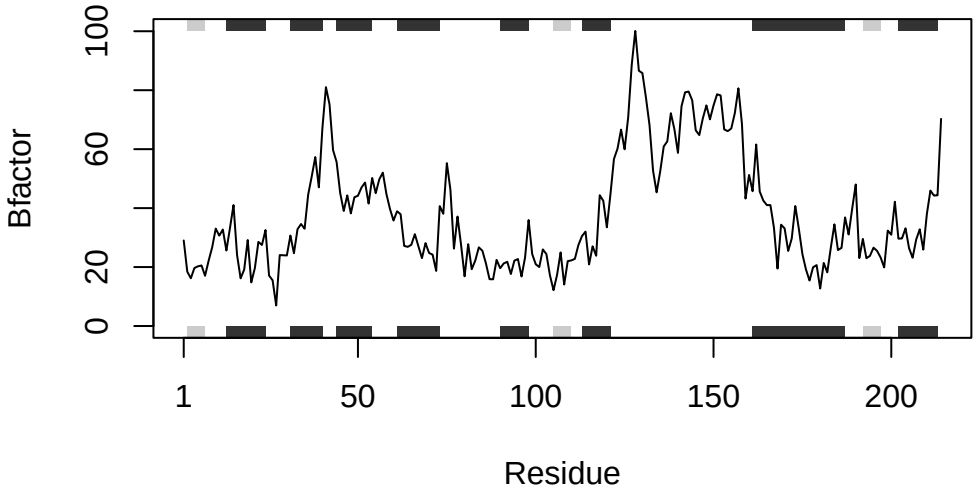
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA") #mistake1: s1 -> s3
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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




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



corrected

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

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE): C:
\Users\zhang\AppData\Local\Temp\Rtmp2HErSz/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\zhang\AppData\Local\Temp\Rtmp2HErSz/1AKE.pdb exists. Skipping download
```

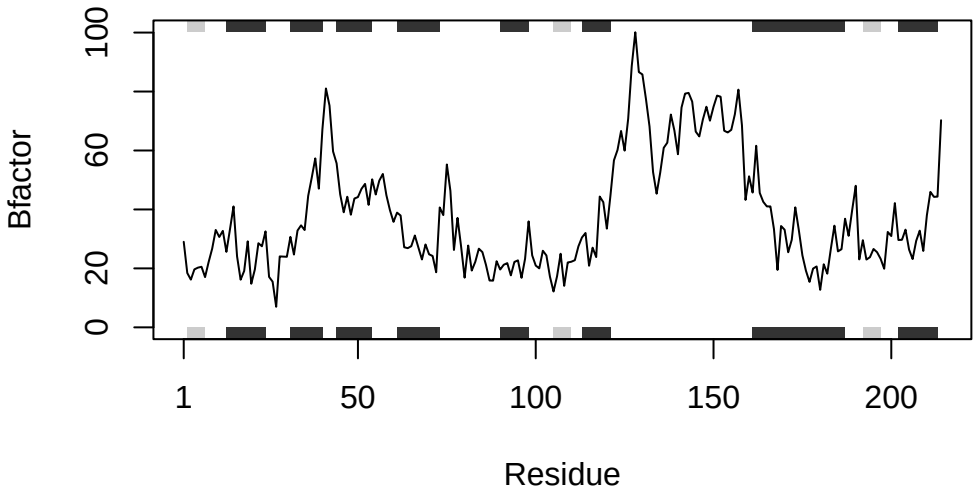
```
   PDB has ALT records, taking A only, rm.alt=TRUE
```

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

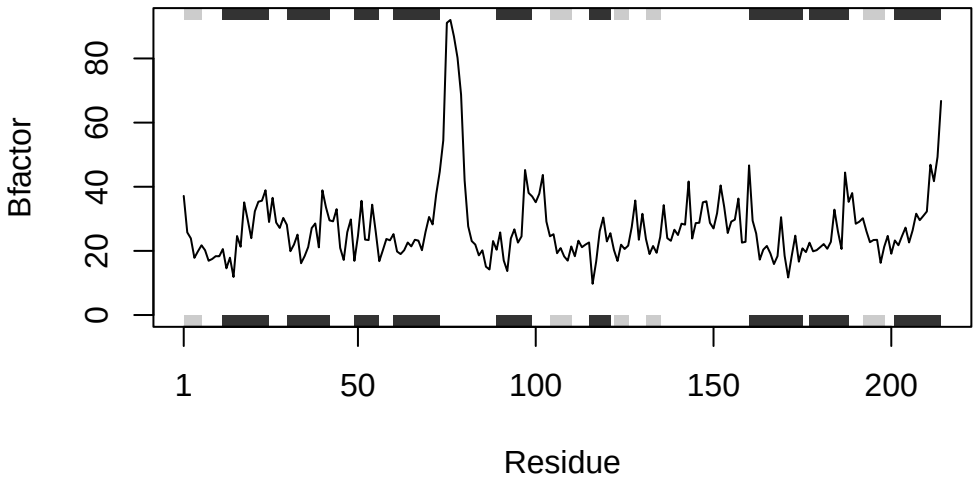
```
  Note: Accessing on-line PDB file
```

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE): C:
\Users\zhang\AppData\Local\Temp\Rtmp2HErSz/1E4Y.pdb exists. Skipping download
```

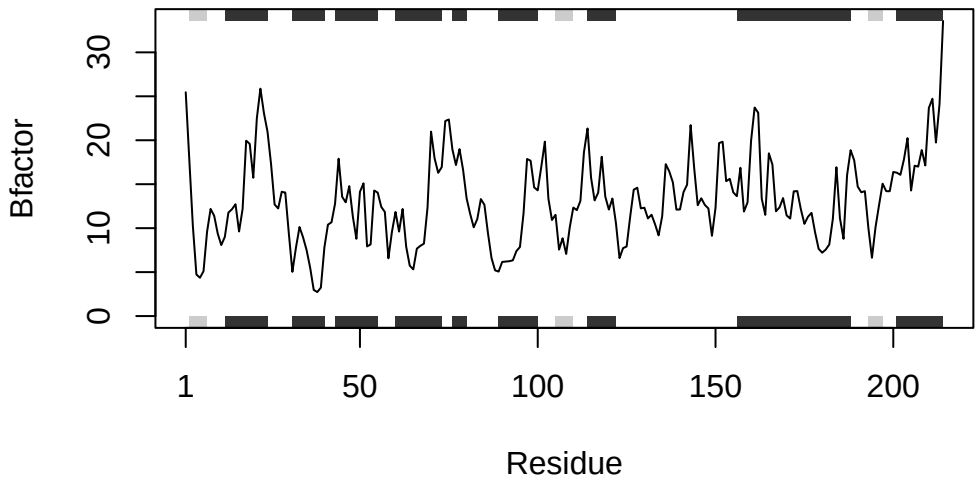
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s3, chain="A", elety="CA") #mistake1: s1 -> s3
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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



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



Student's answer

```
# create a function called plotPDB() which takes in three parameters:
# code: a vector of the PDB codes for the proteins of interest
# chain/elmnt: a string value that indicate the chain/element of interest

plotPDB <- function(code, chain, elmnt){

  color <- seq(20, 120, 100/length(code))  #proteins will then be assigned to different co

  for (i in 1:length(code)){              #use for loop to iterate through each protein co

    s <- read.pdb(code[i])                #import protein information
    s.chain <- trim.pdb(s, chain=chain, elety=elmnt)
    s.b <- s.chain$atom$b                 #trimming, same as the example code

    if (i==1){
      #generate a plot, add the first protein
```

```
      plotb3(s.b, sse=s.chain, typ="l", ylab="Bfactor", col = color[i])
    }else{
      #add lines onto the first plot
      lines(s.b, col = color[i])
    }

  }
  legend("topright", code, fill = color)  #add legend
}
```

test run

```
pdbCode <- c("4AKE", "1AKE", "1E4Y")

color <- c()

plotPDB(pdbCode, "A", "CA")
```

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

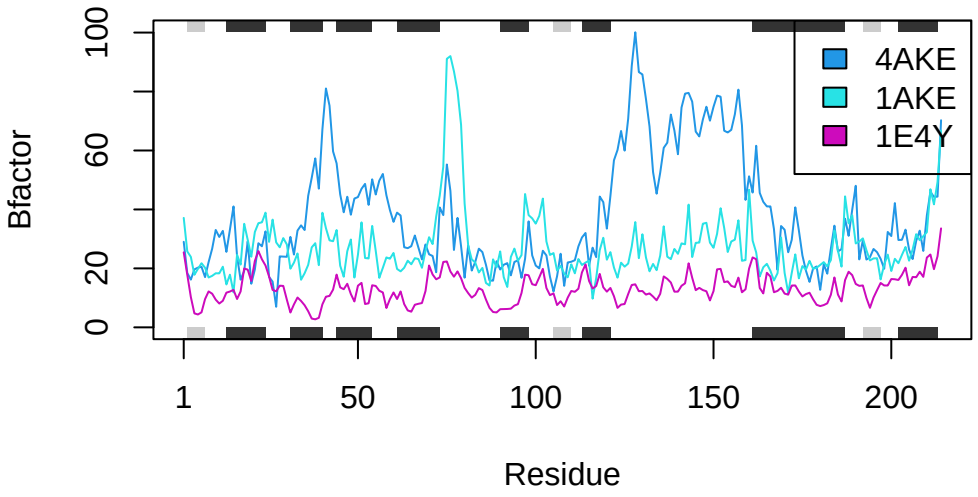
```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE): C:
\Users\zhang\AppData\Local\Temp\Rtmp2HErSz/4AKE.pdb exists. Skipping download
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE): C:
\Users\zhang\AppData\Local\Temp\Rtmp2HErSz/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\zhang\AppData\Local\Temp\Rtmp2HErSz/1E4Y.pdb exists. Skipping download
```


Bfactor
0
20
60
100
1
50
100
150
200
Residue
4AKE
1AKE
1E4Y