Write a function HW

Bangyan Hu (PID: A15540189)

10/17/2021

The original analysis codes:

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

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

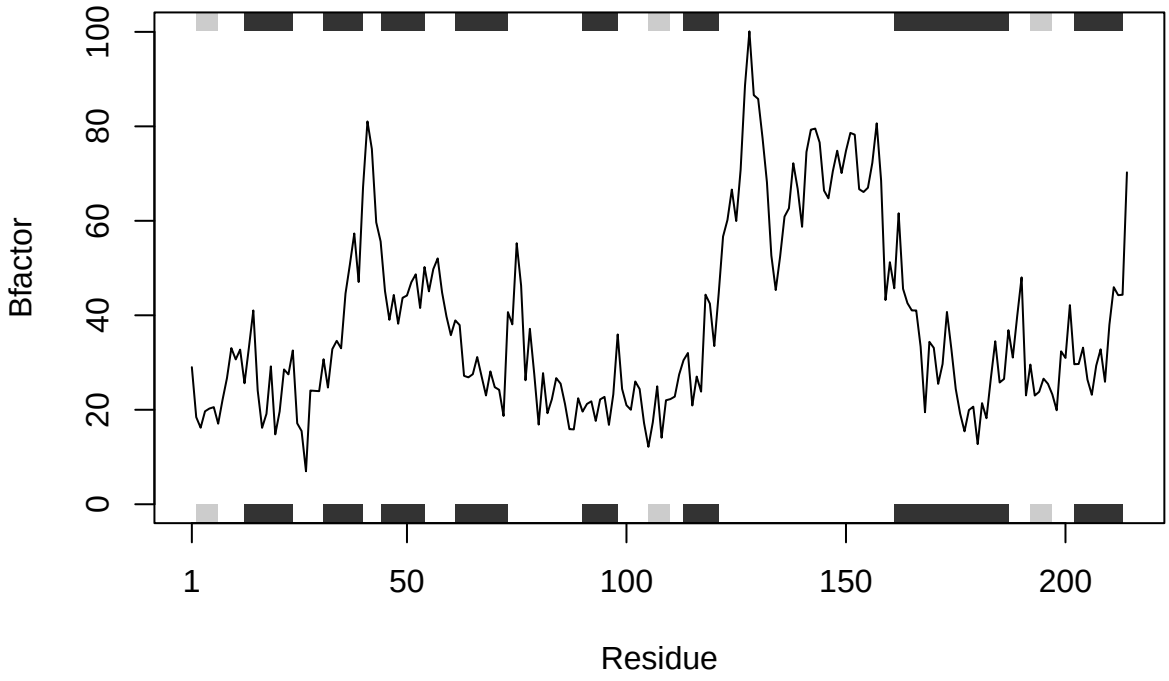
```
s2 <- read.pdb("1AKE") # kinase no drug
```

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

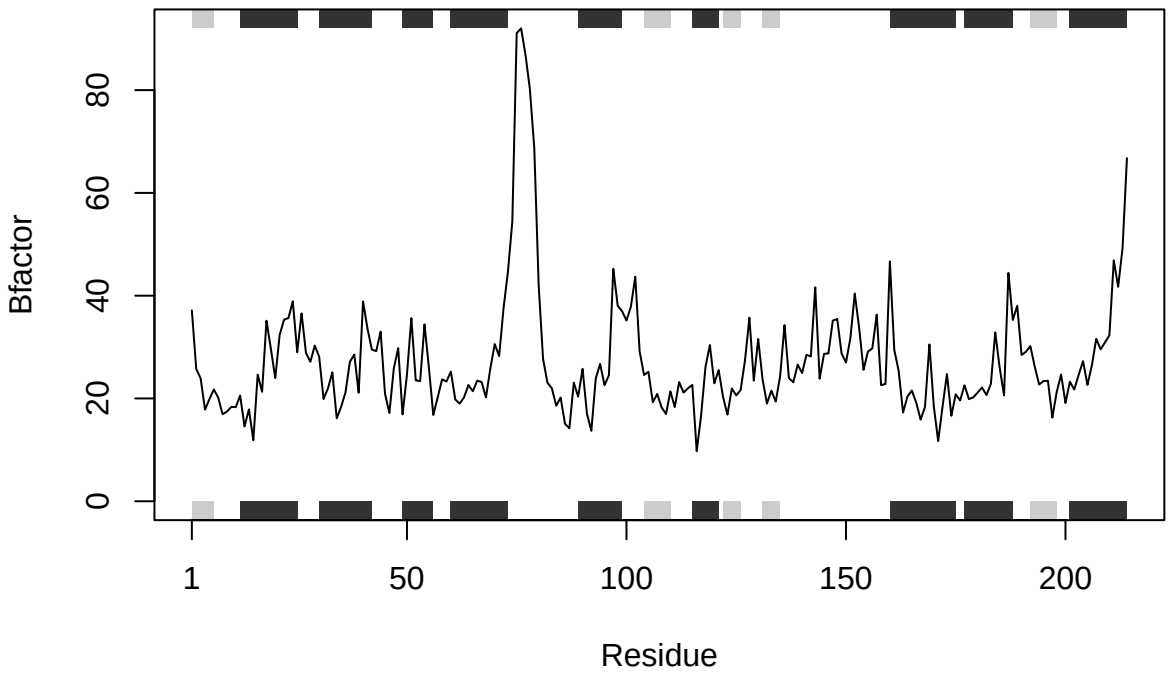
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

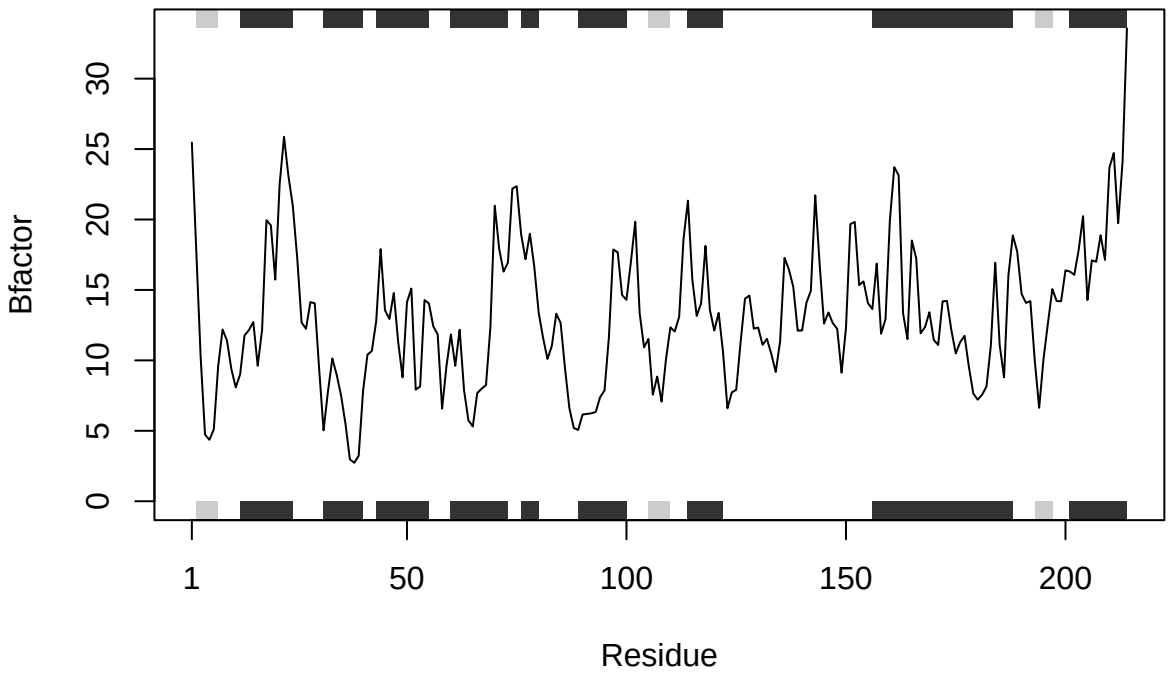
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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




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



Write a function to generalize the original code above to work with any set of input protein structures: prot_drug_plot(File, Fhain, Elmnt). Parameters - File, Chain, Elmnt - are the inputs to the function, which should be inputted as "string".

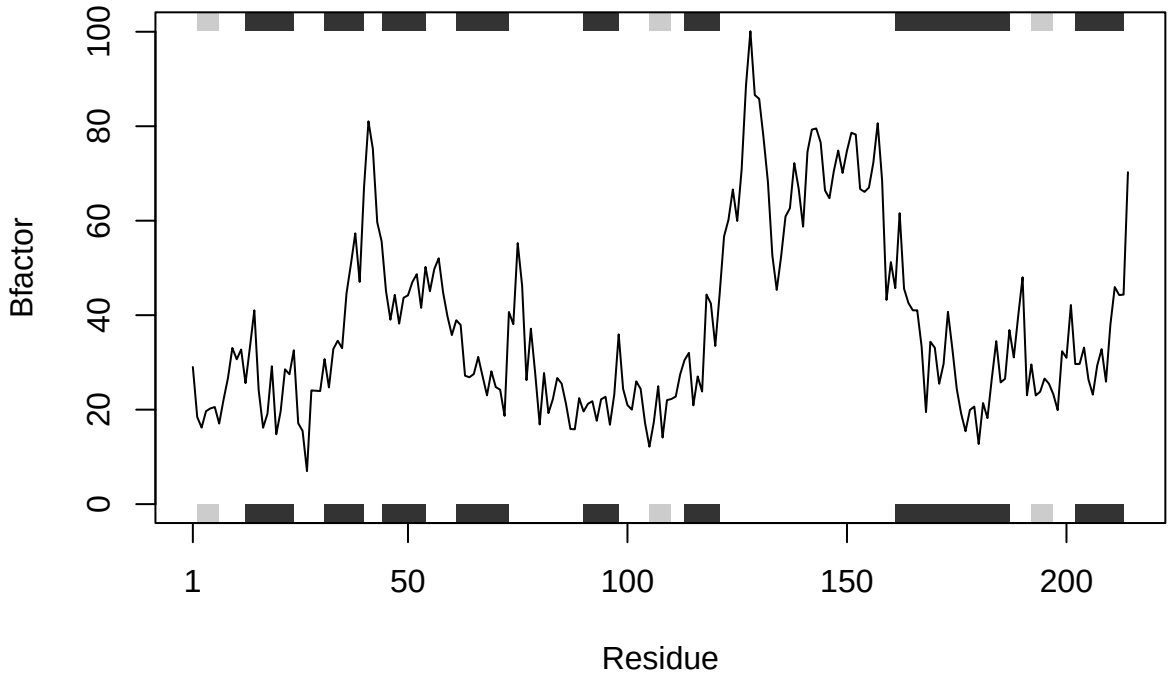
```
#Define the function prot_drug_plot with parameters: File, Chain, Elmnt.
prot_drug_plot <- function(File, Chain, Elmnt) {
  #launch the package bio3d.
  library(bio3d)
  #Generalize the step of reading the file.
  sn <- read.pdb(File)
  #Generalize the step of defining trim.pdb().
  sn.chainN <- trim.pdb(sn, chain = Chain, elety= Elmnt)
  #Generalize the next analytic step.
  sn.b <- sn.chainN$atom$b
  #Generalize the visualization step (generating the plot).
  plotb3(sn.b, sse=sn.chainN, typ="l", ylab="Bfactor")
}
```

The function can be used to generalize the original analysis codes and generate the desired analytic plots. To use this function, prot_drug_plot() should be called with parameters (File, Chain, Elmnt). The output of the function would be the analytic plot (line chart) generated by the parameters inputted.

Then, test the function prot_drug_plot() with three samples given in the original codes:

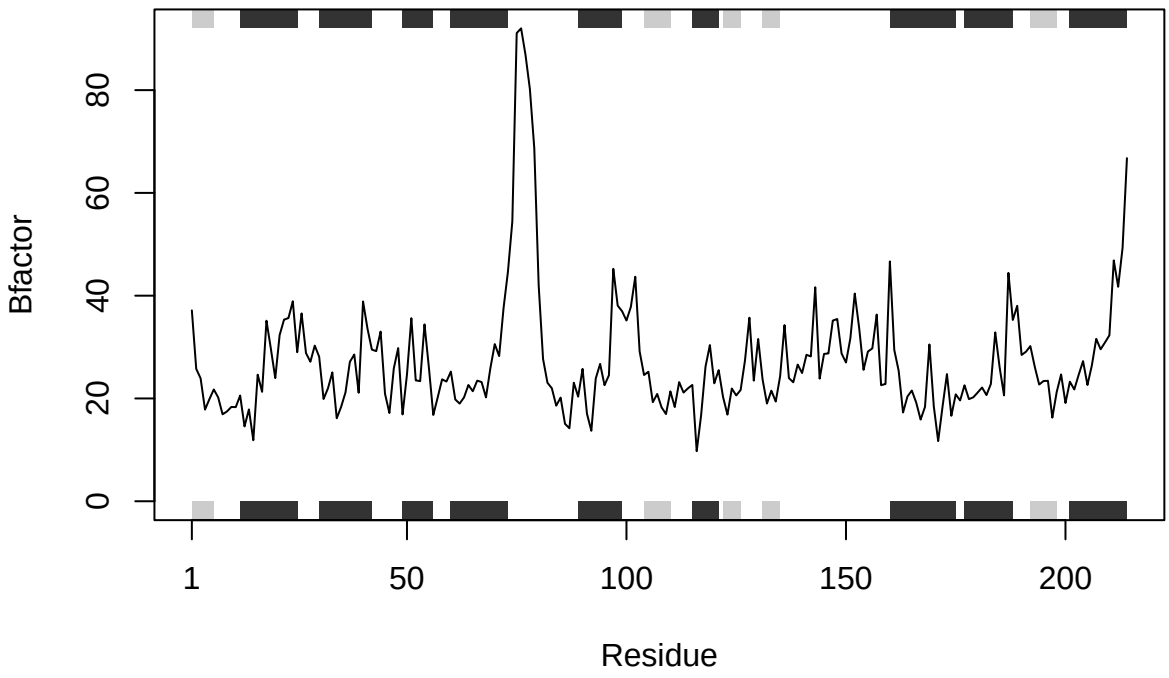
```
#Test the first sample ("4AKE", "A", "CA").
prot_drug_plot("4AKE", "A", "CA")
```

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/
## b0/4hz70r0s5tg0f81tqq9vn2s00000gn/T//Rtmp3rc2Uc/4AKE.pdb exists. Skipping
## download
```



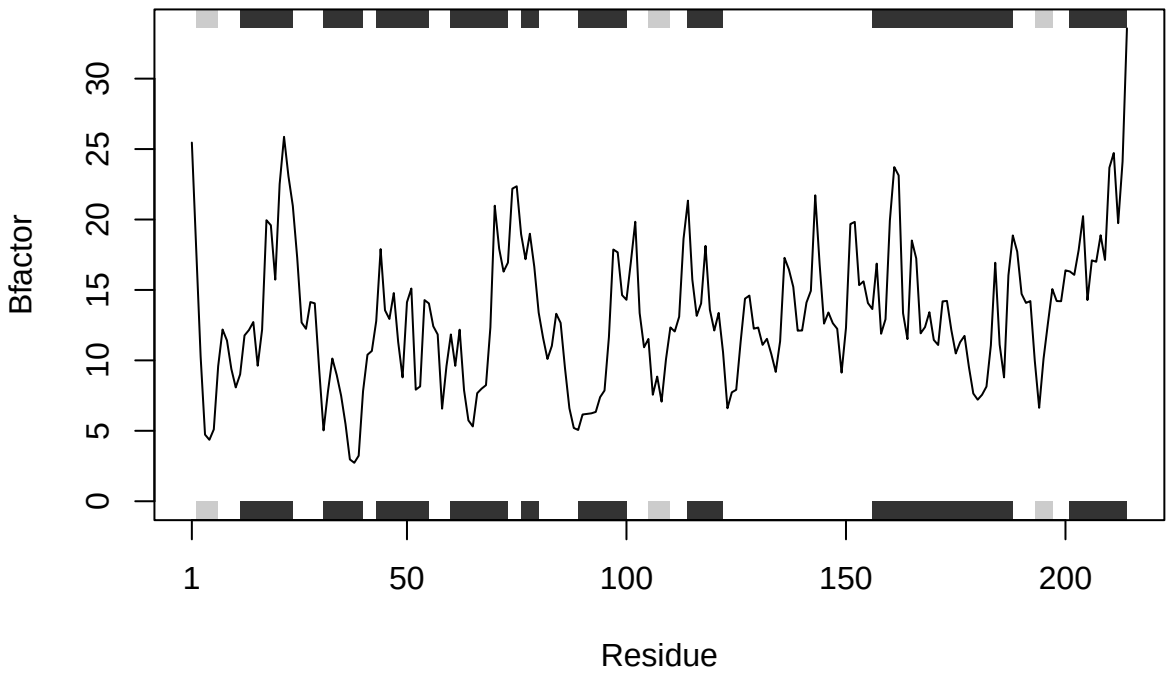

```
#Test the second sample ("1AKE", "A", "CA").
prot_drug_plot("1AKE", "A", "CA")
```

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/
## b0/4hz70r0s5tg0f81tqq9vn2s00000gn/T//Rtmp3rc2Uc/1AKE.pdb exists. Skipping
## download

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




```
#Test the third sample ("1E4Y", "A", "CA").
prot_drug_plot("1E4Y", "A", "CA")
```

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/
## b0/4hz70r0s5tg0f81tqq9vn2s00000gn/T//Rtmp3rc2Uc/1E4Y.pdb exists. Skipping
## download
```



The plots generated by prot_drug_plot() are identical to the original codes, but the use of R function can save lots of time and make the project more organized.