Class 6 Supplementary Homework

2023-10-23

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Here is the code that we are starting with (after we have installed the bio3d package):

```
# Can you improve this analysis code?
library(bio3d)
s1 <- read.pdb("4AKE")  # kinase with drug
```

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

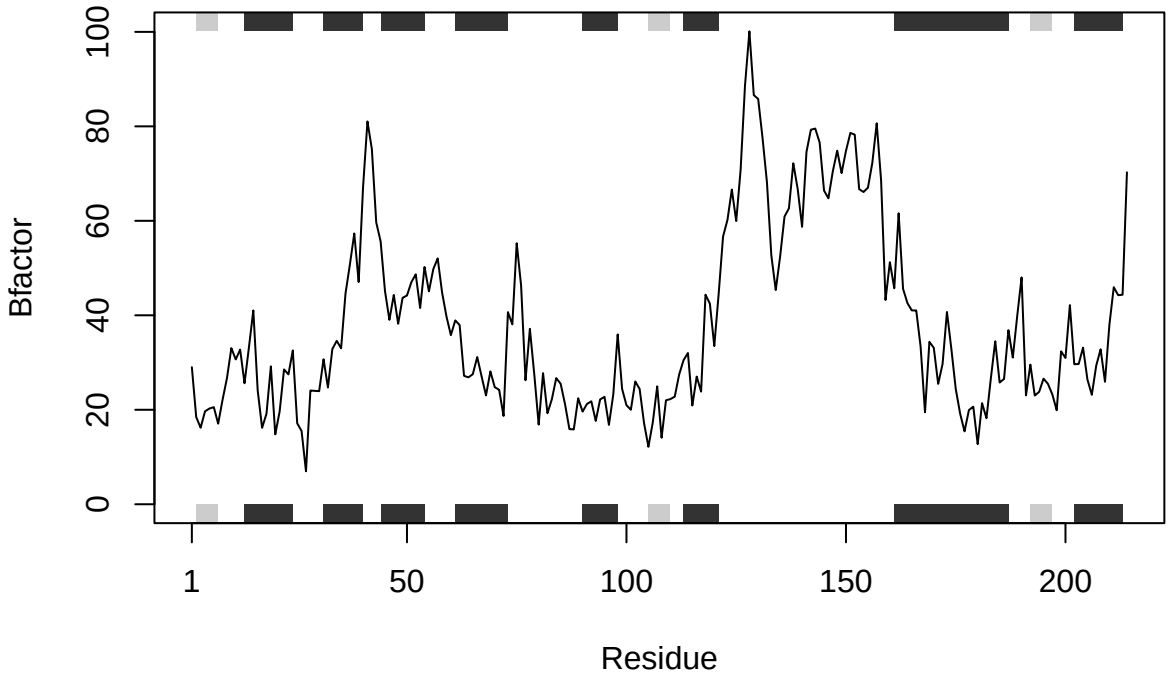
```
s2 <- read.pdb("1AKE")  # kinase no drug
```

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

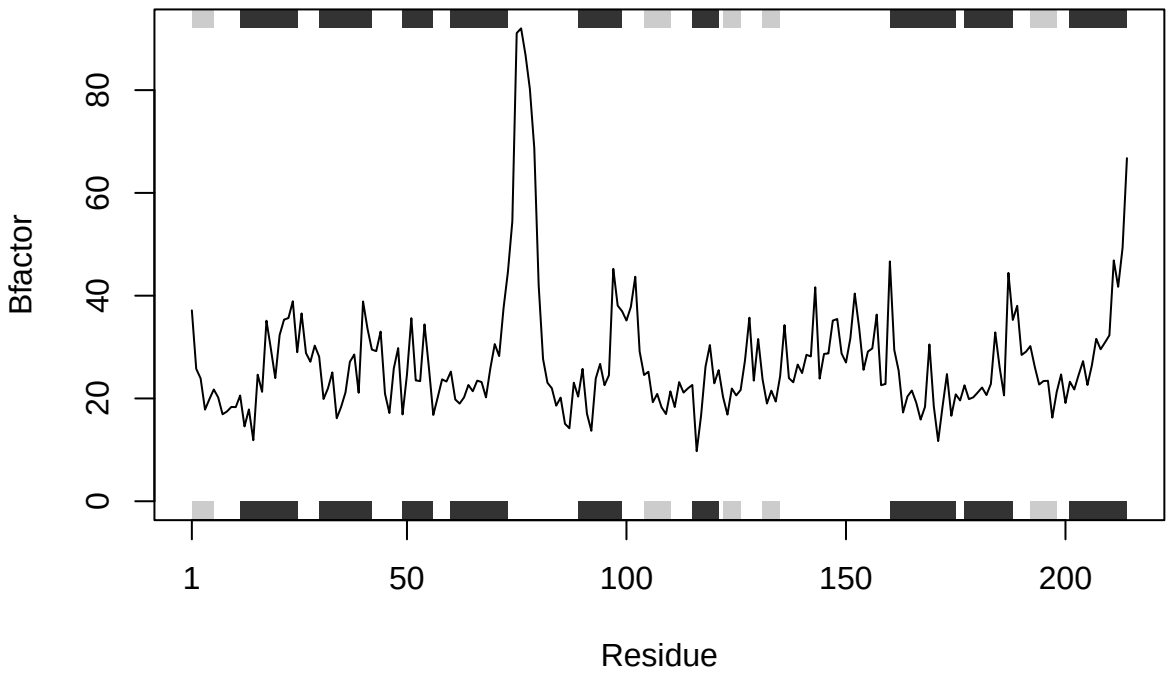
```
s3 <- read.pdb("1E4Y")  # kinase with drug
```

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

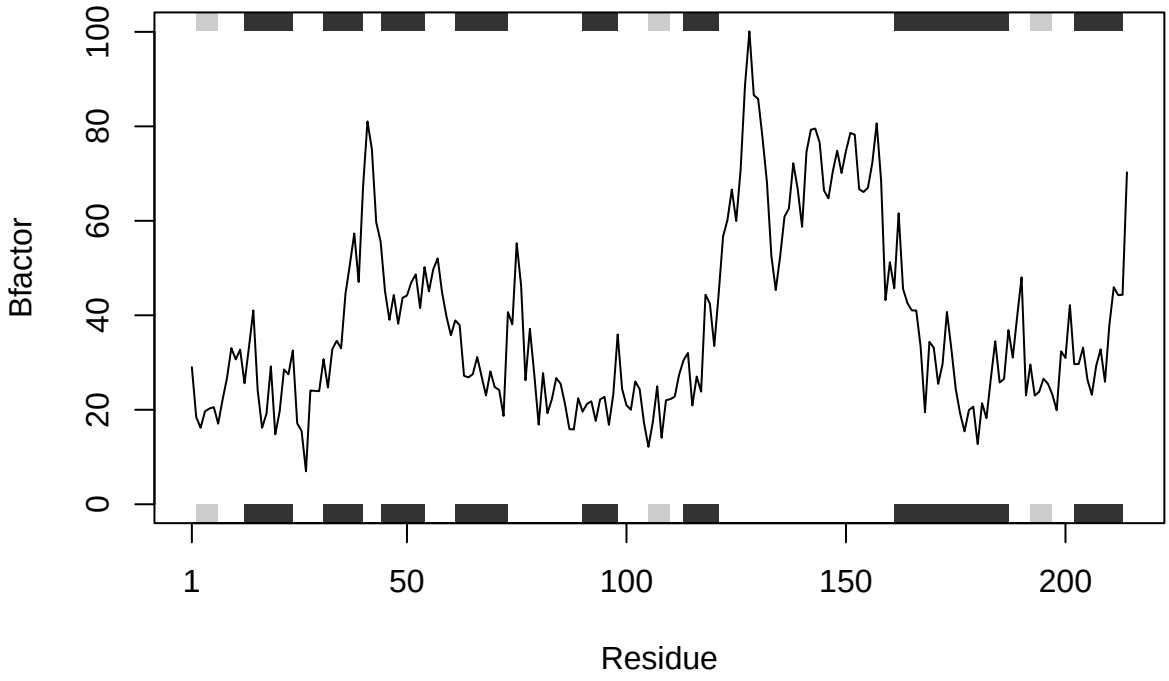
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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




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



It takes our 3 proteins from pdb, and prints out 3 plots using the protein's B factor as a y axis and the residues as an x axis. Due to an error in the code, plot1 and plot3 are identical. Fixing that clerical error will give us 3 separate, unique plots.

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

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

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
## /var/folders/s4/jn1k34tj1wd9x49kbnz6p8sc0000gn/T//Rtmpnjx4FE/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/s4/jn1k34tj1wd9x49kbnz6p8sc0000gn/T//Rtmpnjx4FE/1AKE.pdb exists.
## Skipping download
```

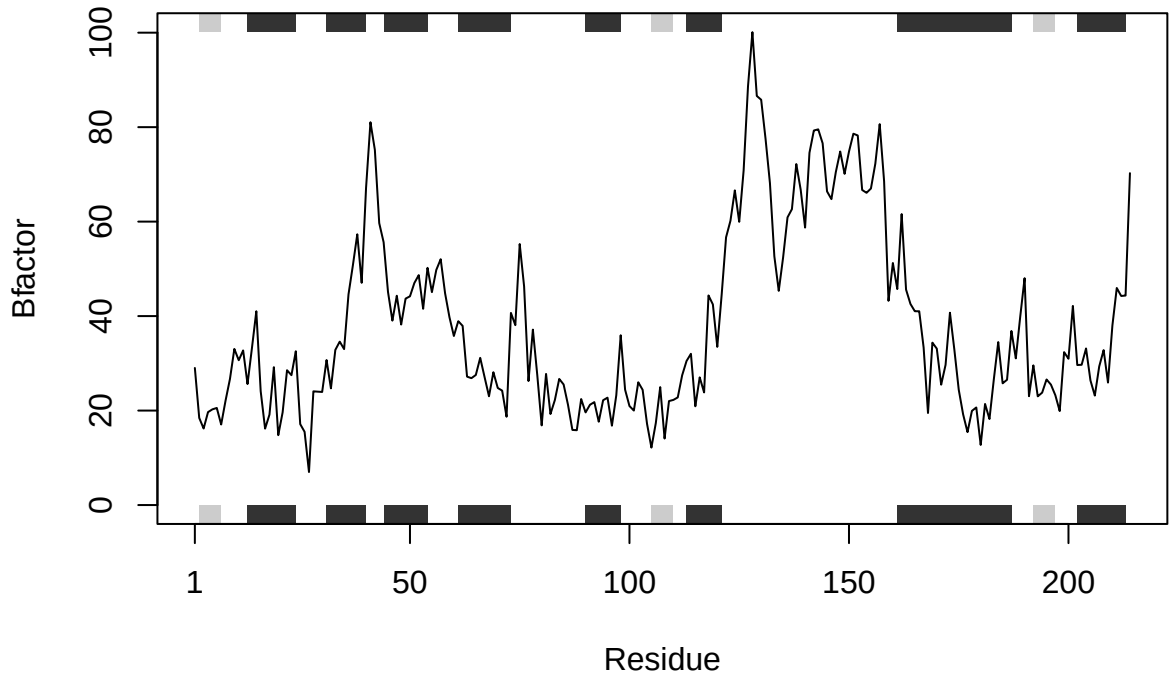
```
##    PDB has ALT records, taking A only, rm.alt=TRUE
```

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

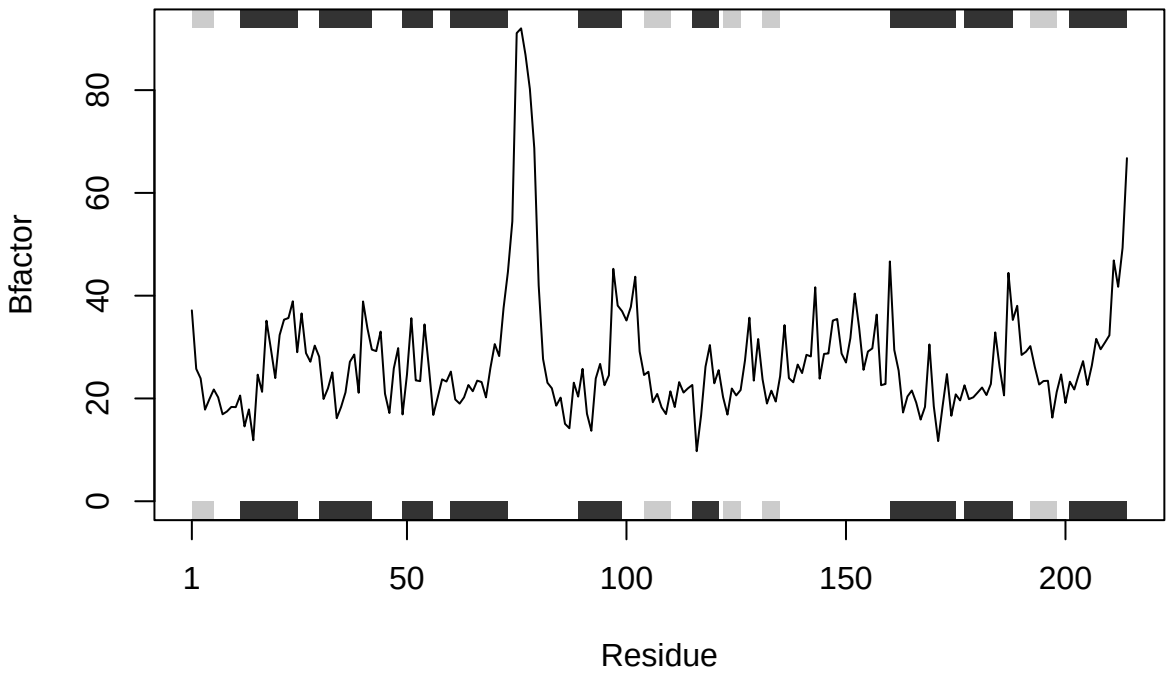
```
##   Note: Accessing on-line PDB file
```

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
## /var/folders/s4/jn1k34tj1wd9x49kbnz6p8sc0000gn/T//Rtmpnjx4FE/1E4Y.pdb exists.
## Skipping download
```

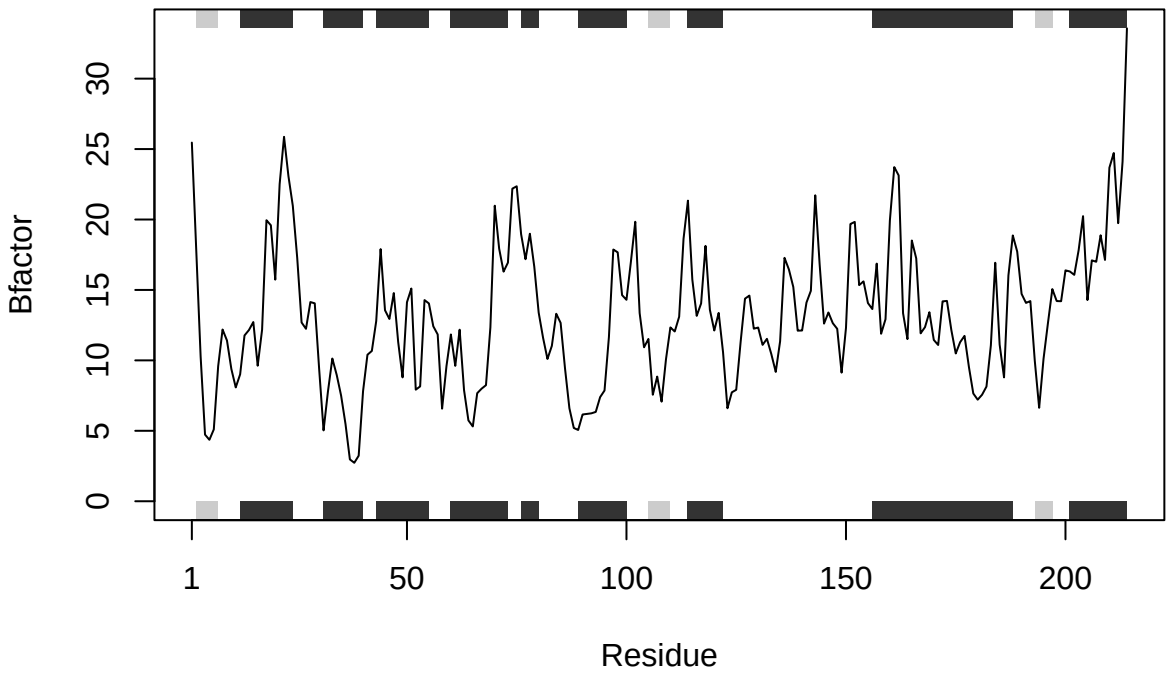
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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




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



To end up with a plot, we need "s1.b" defined, which has been defined in the code above as a trimmed, smaller version of the full pdb structure.

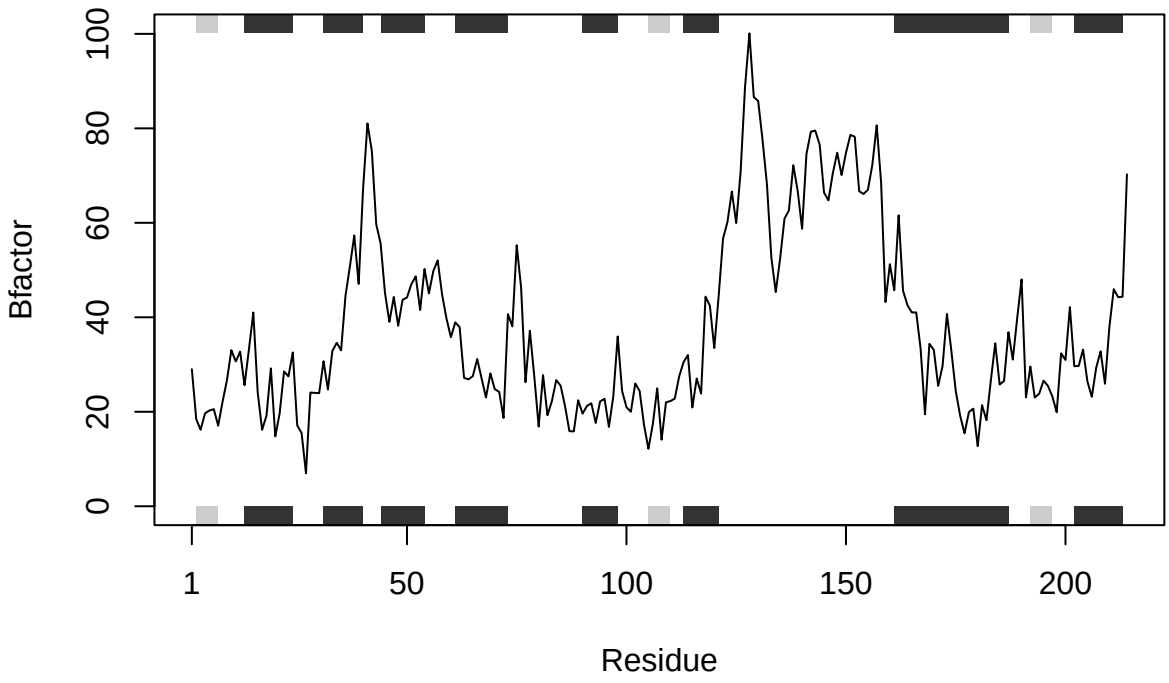
Currently, the program does all the steps to generate the input for the code in 3 repetitions, because it is doing it with 3 separate kinases. We can remove these repeated steps by making one version that has generalized variables that will work with any inputted protein structure.

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

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

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
## /var/folders/s4/jn1k34tj1wd9x49kbnz6p8sc0000gn/T//Rtmpnjx4FE/4AKE.pdb exists.
## Skipping download
```

```
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")
```



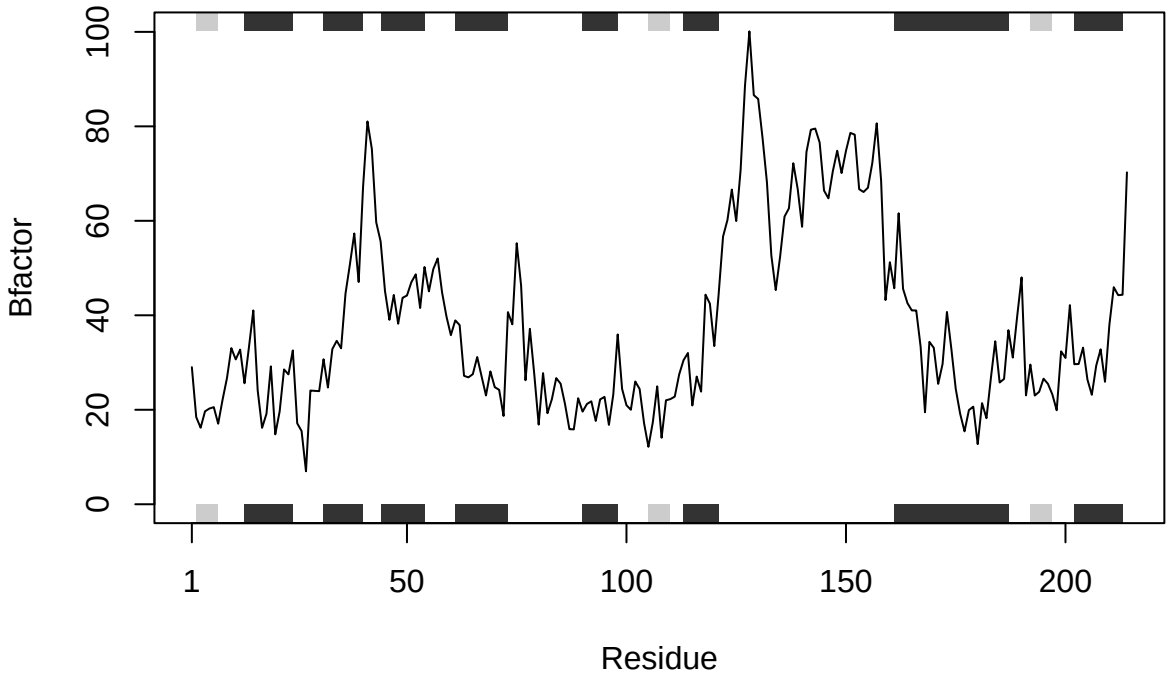
When rewriting, I will redefine new variables with names that are longer but more clear to me than the original function.

```
protein_structure <- read.pdb("4AKE")
```

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

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
## /var/folders/s4/jn1k34tj1wd9x49kbnz6p8sc0000gn/T//Rtmpnjx4FE/4AKE.pdb exists.
## Skipping download
```

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

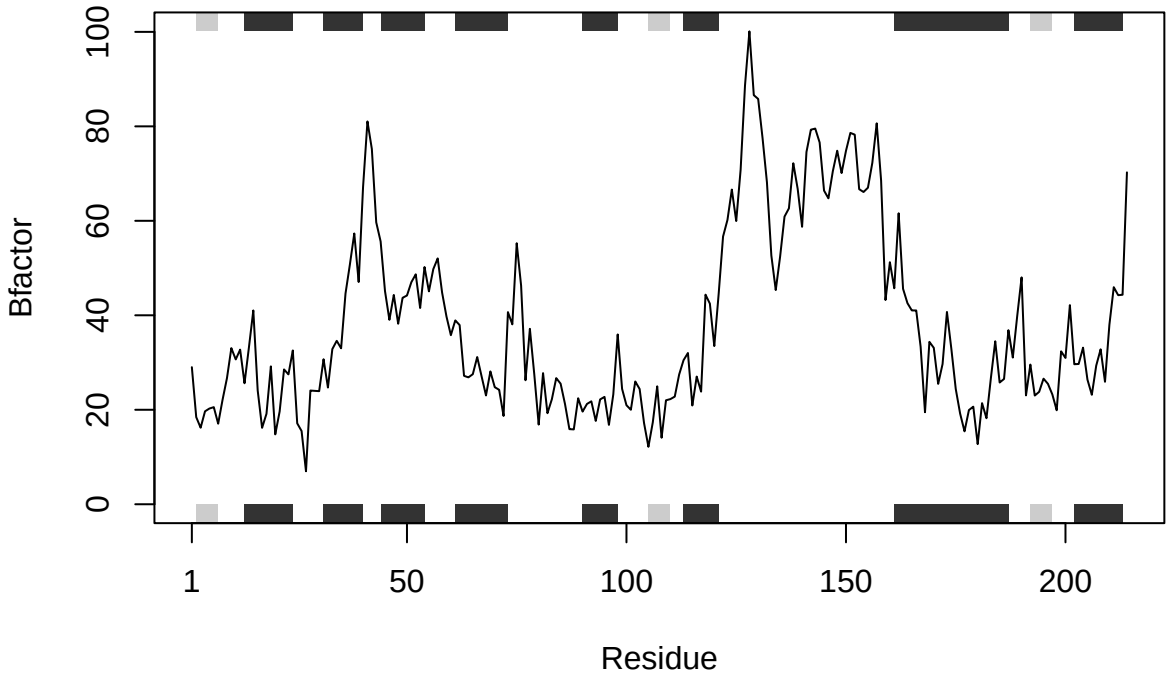


Now, I will put this under one function that only requires the pdb_file variable to be defined in order to run.

```
plot_protein_b_factor <- function(x) {
  protein_structure <- read.pdb(x)
protein_structure_chainA <- trim.pdb(protein_structure, chain="A", elety="CA")
protein_b_factor <- protein_structure_chainA$atom$b
plotb3(protein_b_factor, sse=protein_structure_chainA, typ="l", ylab="Bfactor")
}
## Here's an example
plot_protein_b_factor("4AKE")
```

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

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
## /var/folders/s4/jn1k34tj1wd9x49kbnz6p8sc0000gn/T//Rtmpnjx4FE/4AKE.pdb exists.
## Skipping download
```


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