Class 6: Writing my own Function

Idara: A16865157

Original function

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

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

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

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

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

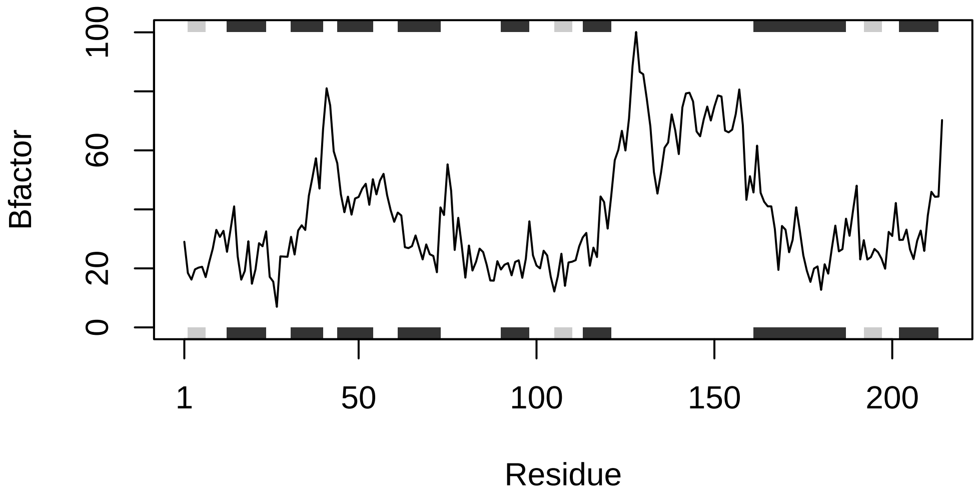
```r
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

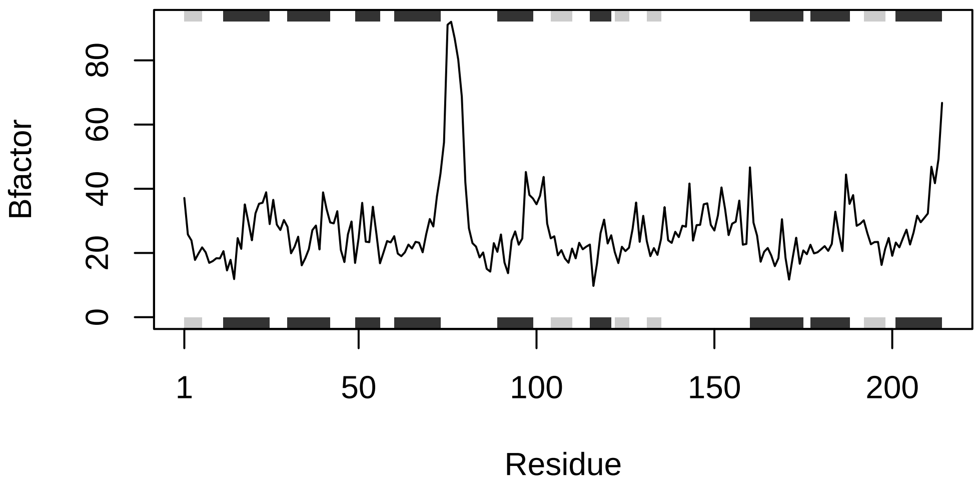
```r
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
```

```r
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
```

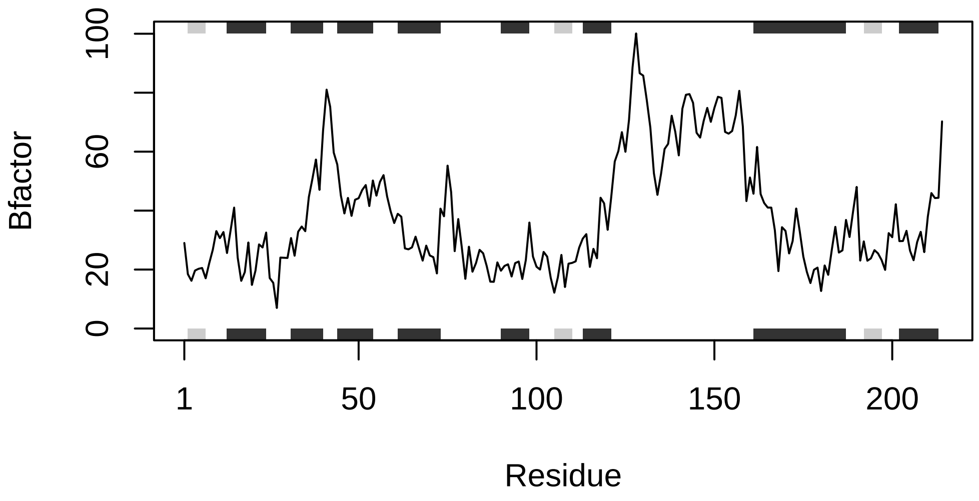
```
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



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



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



I want to simplify the redundancies in this original code and make this succinct yet functional.

```r
install.packages("bio3d")
```

```
Warning: package 'bio3d' is in use and will not be installed
```

Let's make this a little bit less funky! First I want to make sense of the parts so that we can get to a whole. So let's start with the pdb file

##Inputs of the Function

In order to read and understand this package, I am going to import `bio3d()` so that we can read the pdb file our input

```r
library(bio3d)
```

Now that this is done I want to define the pdb file using `function(){}` and then store the resultant output in `plot_b_factors`. Recall how to structure a function

```r
plot_protein <- function(pdb_file){
  library(bio3d)
}
```

Next, read the pdb file

```r
plot_protein <- function(pdb_file){
  library(bio3d)
pdb_data <- read.pdb(pdb_file)}
```

Then trim the pdb structure to zer in on data we want

```r
plot_protein <- function(pdb_file) {
  library(bio3d)
  pdb_data <- read.pdb(pdb_file)
  pdb_chain <- trim.pdb(pdb_data, chain="A", elety="CA")
}
```

Next we want to specifically identify correct information from dataset to plot

```r
plot_protein <- function(pdb_file) {
  library(bio3d)
  #read the pdb data
  pdb_data <- read.pdb(pdb_file)
  #make a subset of the specific protein data we want
  pdb_chain <- trim.pdb(pdb_data, chain="A", elety="CA")
  #call correct info from the data to plot
  bfactors <- pdb_chain$atom$b
}
```

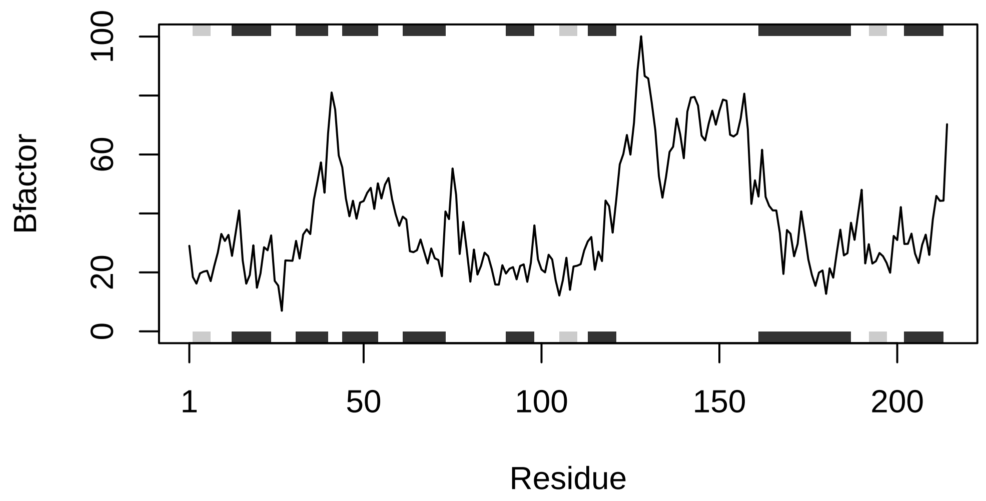
Next, we want to plot B-factor values using `plotb3()`.

```r
plot_protein <- function(pdb_file) {
  library(bio3d)
  pdb_data <- read.pdb(pdb_file)
  #read the pdb data
  pdb_chain <- trim.pdb(pdb_data, chain="A", elety="CA")
  #make a subset of the specific protein data we want
  bfactors <- pdb_chain$atom$b
  #call correct info from the data to plot
  plotb3(bfactors, sse=pdb_chain, typ="l",   ylab="Bfactor")
}

plot_protein("4AKE")
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\idara\AppData\Local\Temp\RtmpmI7V4r/4AKE.pdb exists. Skipping download
```



Overall this function of `plot_protein`takes the `pdb_file` argument to define and creates an approved path to the pdb file. This reads the pdb file, trims the file to include chain “A” and elety=“CA” , extracts B-factor values and then proceeds to plot them.