class 6 hw

Trinity Leahy

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

This is the original code:

```
# 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
```

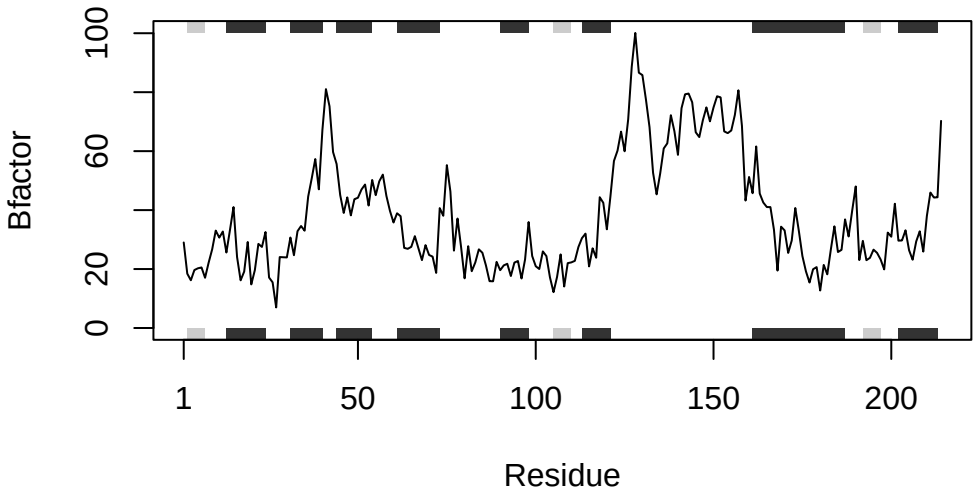
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

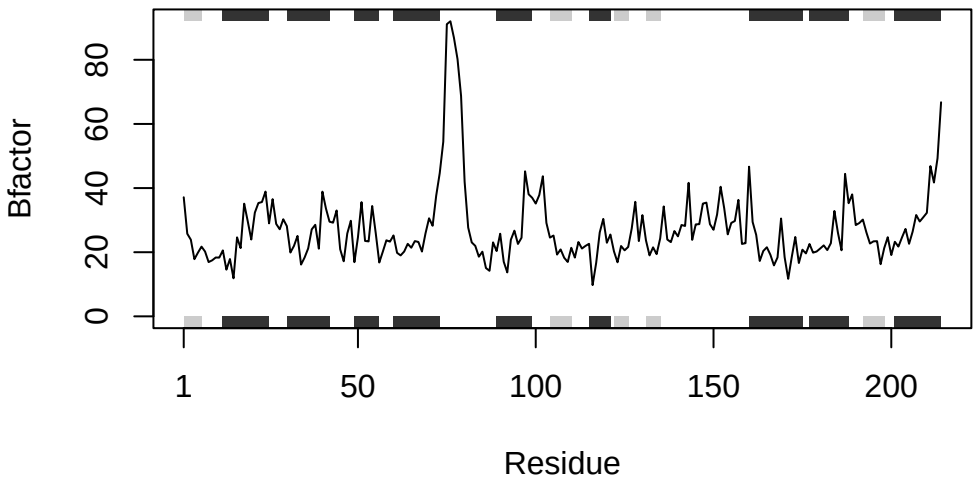
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")

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

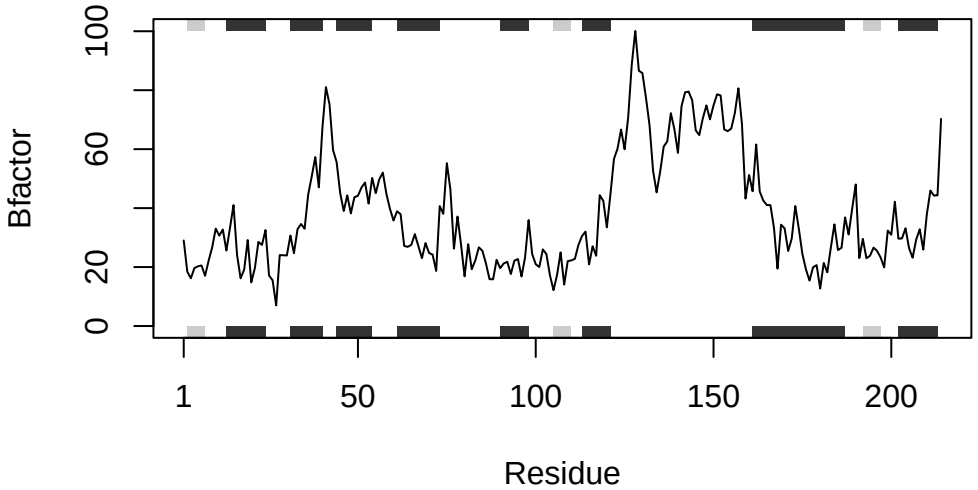
```
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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




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



The Breakdown

To figure out how to simplify the code and turn it into a function, I will become by breaking down each code chunk so I can understand what it accomplishes.

I will begin by calling package `bio3d` up from my library.

```
library(bio3d)
```

Next, I am going to break down each code structure to determine what they do and if they are functional.

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

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/z3/bbbbbfqn61qc1yx67hrd0l4m0000gn/T//RtmpxEJQUy/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/z3/bbbbbfqn61qc1yx67hrd0l4m0000gn/T//RtmpxEJQUy/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/z3/bbbbbfqn61qc1yx67hrd0l4m0000gn/T//RtmpxEJQUy/1E4Y.pdb exists.
Skipping download
```

The above lines of code use the unique PDB accession codes in order to access a file about a specific protein.

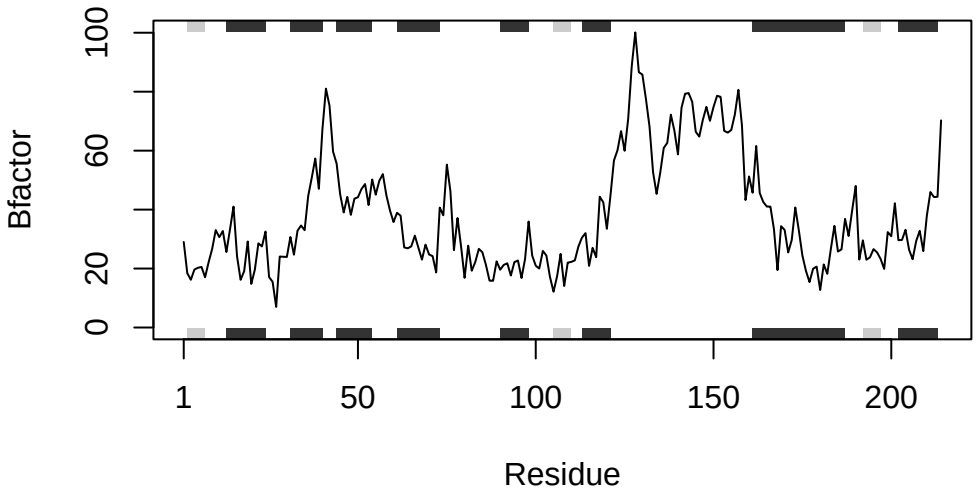
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")
```

The `trim.pdb()` function calls on R to access a specific chain within the protein we are interested in. The input arguments are (protein (defined by pdb), chain (defined by what chain we are interested in), and electy (character vector of atom names)).

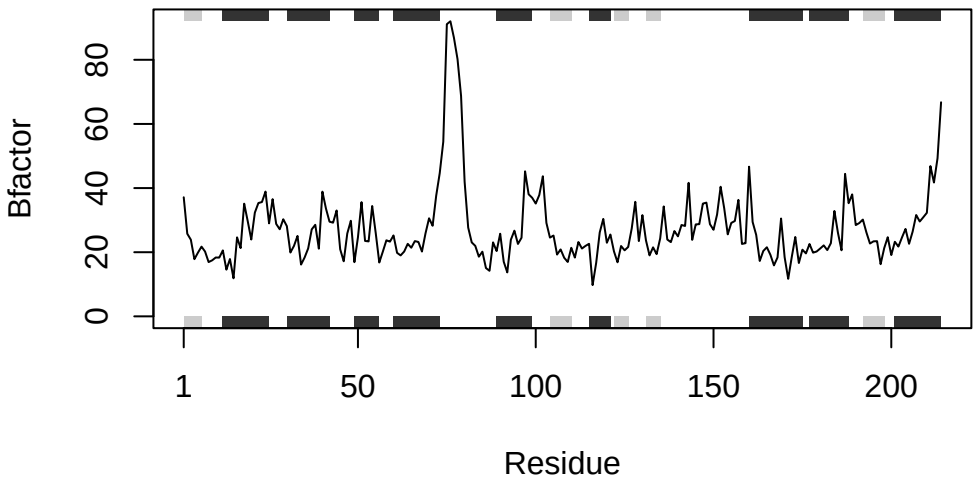
```
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
```

This code even further specifies the part of the protein we are looking at.

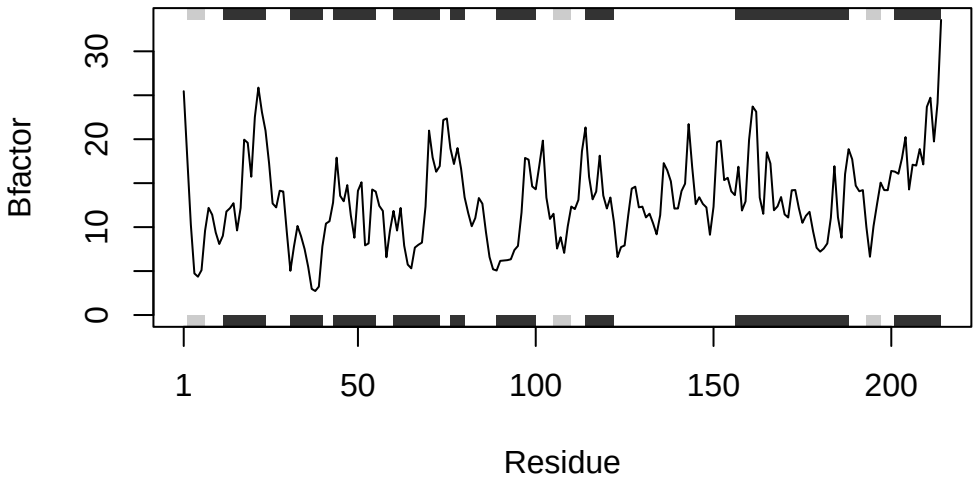
```
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```




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




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



This code plots the information we are looking at pertaining to our protein, with B-Factor on the Y-axis and Residue on the X-axis.

Simplifying

Now I am going to simplify the code so that it is more straight forward and concise.

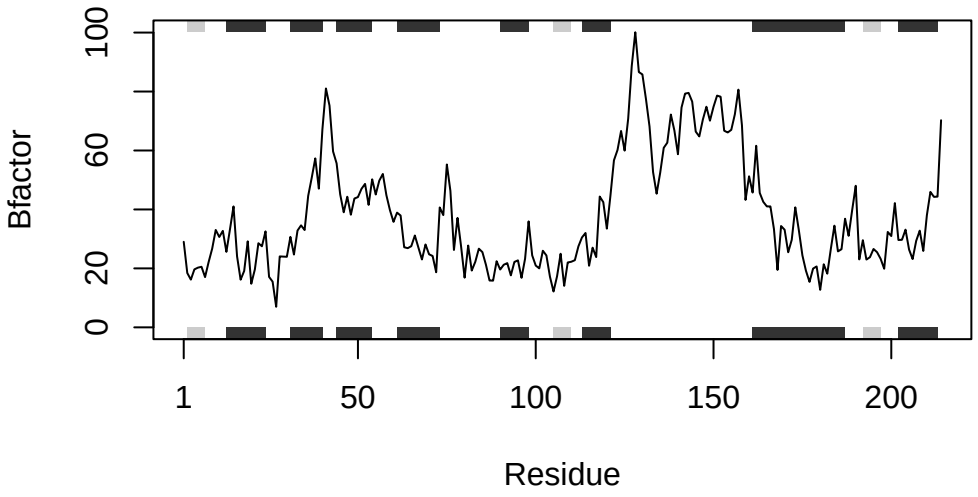
```
#calling forward the package bio3d
library(bio3d)

#specifying what protein and which aspect it we want to examine
s1.chainA <- trim.pdb(read.pdb("4AKE"), chain="A", elety="CA")
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/z3/bbbbbfqn61qc1yx67hrd0l4m0000gn/T//RtmpxEJQUy/4AKE.pdb exists.
Skipping download
```

```
#plotting the data for the specific part of the protein
plotb3(s1.chainA$atom$b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



Function Creation

Finally, I am going to transfer my more concise code into a function.

```
examine.protein <- function(x){
#calling forward the package bio3d
library(bio3d)

#specifying what protein and which aspect it we want to examine
chainA <- trim.pdb(read.pdb(x), chain="A", elety="CA")

#plotting the data for the specific part of the protein
plotb3(chainA$atom$b, sse=chainA, typ="l", ylab="Bfactor")}
```

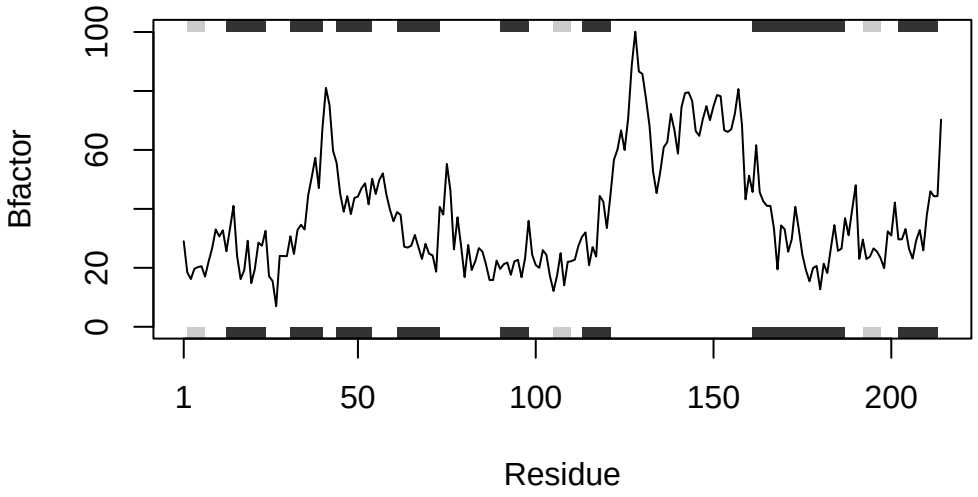
And now we check to make sure that the function does what we are looking for!

```
examine.protein("4AKE")
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
```

```
/var/folders/z3/bbbbbfqn61qc1yx67hrd0l4m0000gn/T//RtmpxEJQUy/4AKE.pdb exists.
Skipping download
```

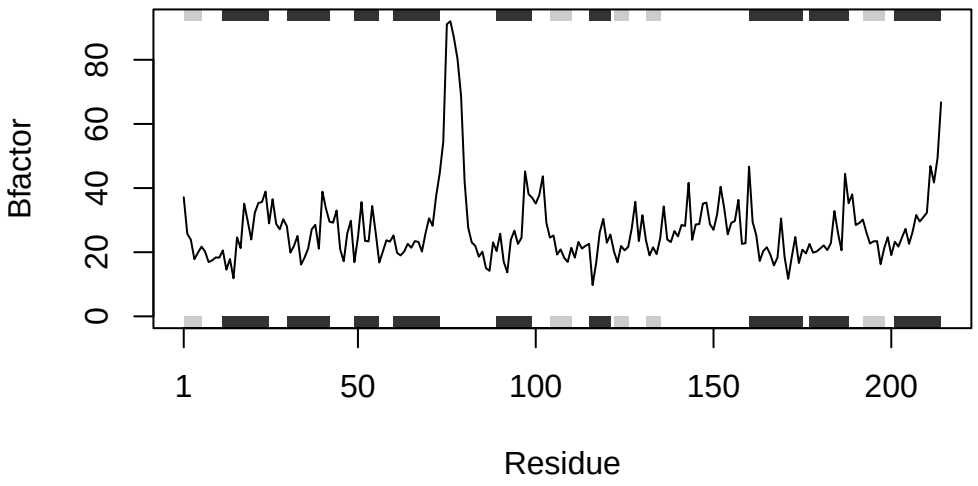



```
examine.protein("1AKE")
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/z3/bbbbbfqn61qc1yx67hrd0l4m0000gn/T//RtmpxEJQUy/1AKE.pdb exists.
Skipping download
```

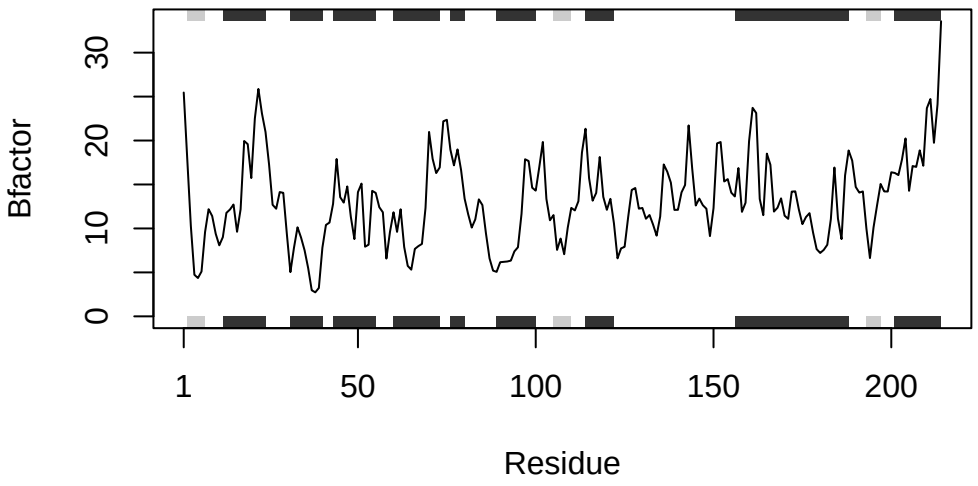
```
   PDB has ALT records, taking A only, rm.alt=TRUE
```




```
examine.protein("1E4Y")
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/z3/bbbbbfqn61qc1yx67hrd0l4m0000gn/T//RtmpxEJQUy/1E4Y.pdb exists.
Skipping download
```



The `examine.protein()` function I created takes the **input** of a PDB code and specifies the part of the protein we are looking at to be a specific part of chain A and then **outputs** a plot of the B-Factor vs Residue data for that part of the protein. It can be used for any protein with a PDB code and multiple chains to look at the chain A of the protein.