R functions

Jiachen Fan (A17662703)

A. Improve this regular R code by abstracting the main activities in your own new function. Note, we will go through this example together in the formal lecture. The main steps should entail running through the code to see if it works, simplifying to a core working code snippet, reducing any calculation duplication, and finally transferring your new streamlined code into a more useful function for you.

Let's have a look at original code

```
df <- data.frame(a=1:10, b=seq(200,400,length=10),c=11:20,d=NA)
df
```

```
    a        b  c  d
1   1 200.0000 11 NA
2   2 222.2222 12 NA
3   3 244.4444 13 NA
4   4 266.6667 14 NA
5   5 288.8889 15 NA
6   6 311.1111 16 NA
7   7 333.3333 17 NA
8   8 355.5556 18 NA
9   9 377.7778 19 NA
10 10 400.0000 20 NA
```

```
df$a <- (df$a - min(df$a)) / (max(df$a) - min(df$a)) # normalize values in df$a
df
```

```
           a        b  c  d
1  0.0000000 200.0000 11 NA
2  0.1111111 222.2222 12 NA
3  0.2222222 244.4444 13 NA
4  0.3333333 266.6667 14 NA
5  0.4444444 288.8889 15 NA
6  0.5555556 311.1111 16 NA
7  0.6666667 333.3333 17 NA
8  0.7777778 355.5556 18 NA
9  0.8888889 377.7778 19 NA
10 1.0000000 400.0000 20 NA
```

```
df$b <- (df$b - min(df$a)) / (max(df$b) - min(df$b))
df
```

```
           a        b  c  d
1  0.0000000 1.000000 11 NA
2  0.1111111 1.111111 12 NA
3  0.2222222 1.222222 13 NA
4  0.3333333 1.333333 14 NA
5  0.4444444 1.444444 15 NA
6  0.5555556 1.555556 16 NA
7  0.6666667 1.666667 17 NA
8  0.7777778 1.777778 18 NA
9  0.8888889 1.888889 19 NA
10 1.0000000 2.000000 20 NA
```

```
df$c <- (df$c - min(df$c)) / (max(df$c) - min(df$c)) # normalize values in df$c
df
```

```
           a        b         c  d
1  0.0000000 1.000000 0.0000000 NA
2  0.1111111 1.111111 0.1111111 NA
3  0.2222222 1.222222 0.2222222 NA
4  0.3333333 1.333333 0.3333333 NA
5  0.4444444 1.444444 0.4444444 NA
6  0.5555556 1.555556 0.5555556 NA
7  0.6666667 1.666667 0.6666667 NA
8  0.7777778 1.777778 0.7777778 NA
9  0.8888889 1.888889 0.8888889 NA
10 1.0000000 2.000000 1.0000000 NA
```

```
df$d <- (df$d - min(df$d)) / (max(df$a) - min(df$d))
df
```

```
           a        b         c  d
1  0.0000000 1.000000 0.0000000 NA
2  0.1111111 1.111111 0.1111111 NA
3  0.2222222 1.222222 0.2222222 NA
4  0.3333333 1.333333 0.3333333 NA
5  0.4444444 1.444444 0.4444444 NA
6  0.5555556 1.555556 0.5555556 NA
7  0.6666667 1.666667 0.6666667 NA
8  0.7777778 1.777778 0.7777778 NA
9  0.8888889 1.888889 0.8888889 NA
10 1.0000000 2.000000 1.0000000 NA
```

Now we can simplify it using function 'normalize'

```
# Define a normalization function
normalize <- function(x) {
    (x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE))
}

# Create data frame
df <- data.frame(a=1:10, b=seq(200,400,length=10),c=11:20,d=NA)

# Apply normalize to df$a and df$c
df[c('a','c')] <- apply(df[c('a','c')], 2, normalize)
df$b <- normalize(df$b)+1
#(df$b - min(df$a)) / (max(df$b) - min(df$b)) = (df$b - min(df$b)+min(df$b)-min(df$a)) / (
df$d <- (df$d - min(df$d)) / (max(df$a) - min(df$d))

# It is hard to improve because the codes are different for each column. It will be much e
```

B. Next improve the below example code for the analysis of protein drug interactions by abstracting the main activities in your own new function. Then answer questions 1 to 6 below. It is recommended that you start a new Project in RStudio in a new directory and then install the bio3d package noted in the R code below (N.B. you can use the command install.packages("bio3d") or the RStudio interface to do this). Then run through the code to see if it works, fix any copy/paste errors before simplifying to a core working code snippet, reducing any calculation duplication, and finally transferring it into a more useful function for you.

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

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

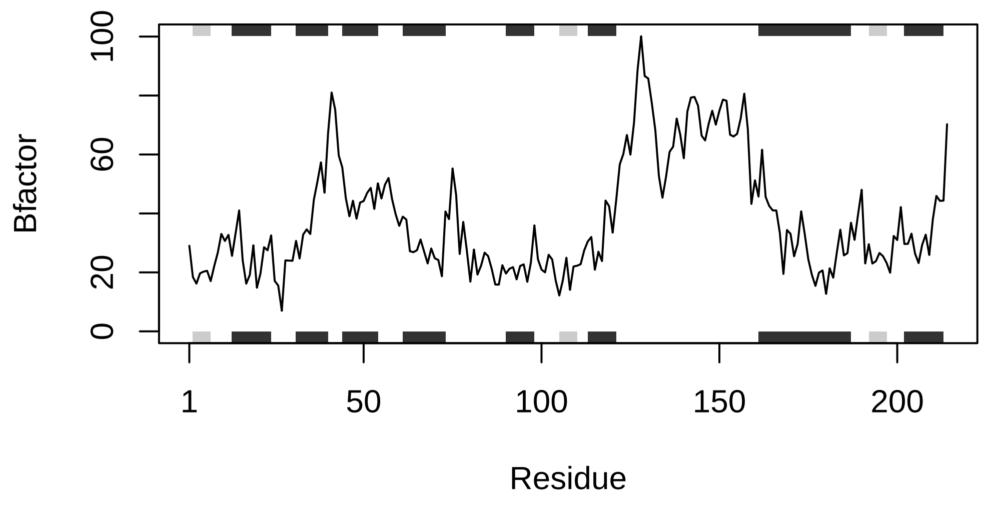
```
s2 <- read.pdb("1AKE") # kinase no drug
```

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

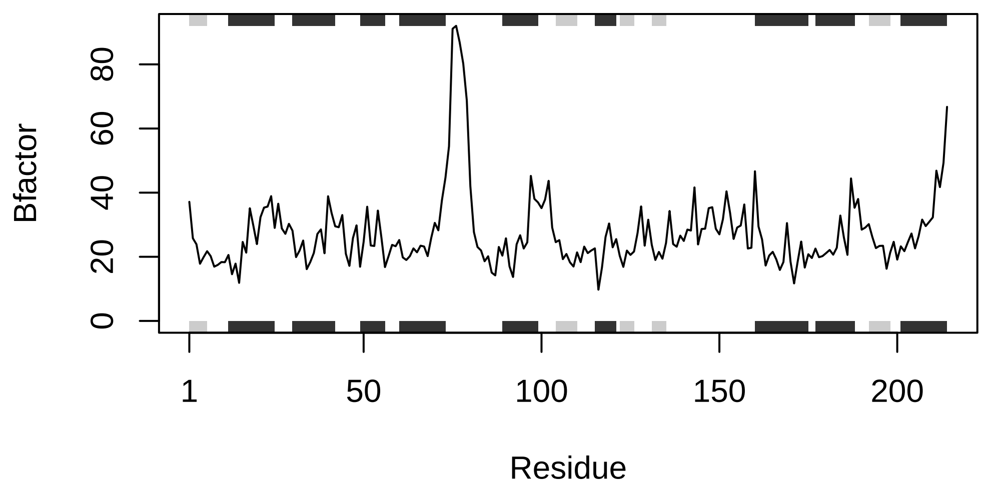
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

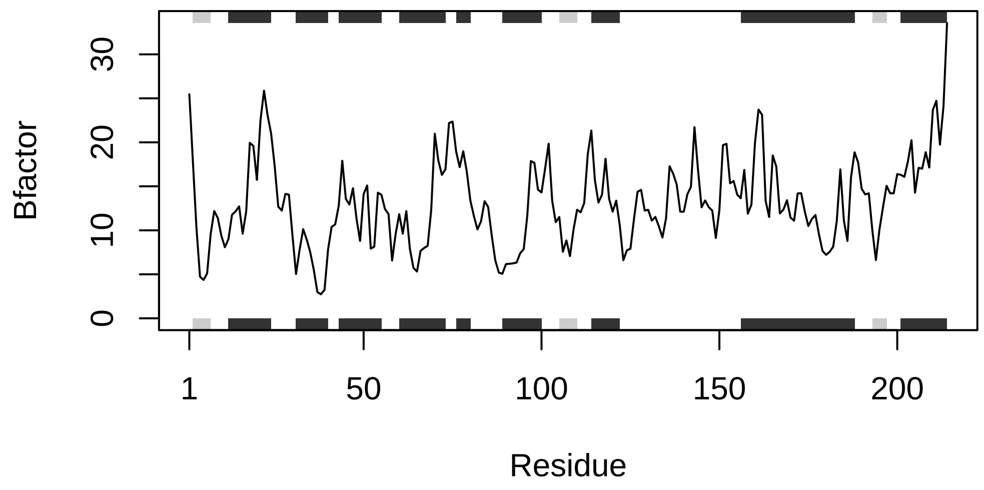
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")         # change 's1' to 's3'
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")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



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



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



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

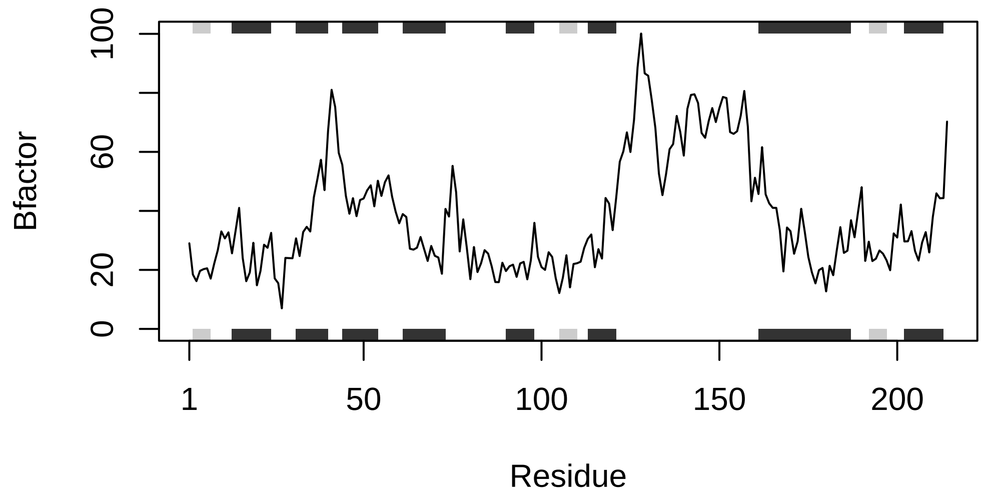
Function 'Bfactor' can receive any protein PDB data, and return a plot of Bfactor. It receives the four-letter PDB identifier (PDBID) and a chain identifier (chainID) as its inputs, and plots the temperature factors (B factors) of the backbone C-alpha atoms in a specified chain of the protein.

```r
# First input install.packages("bio3d") in console

# Generate function 'Bfactor'
library(bio3d)
Bfactor <- function(PDBID, chainID){
  x <- read.pdb(PDBID)                                     # Read the PDB data
  x.chain <- trim.pdb(x, chain=chainID, elety="CA")        # Trim the structure to get only
  x.b <- x.chain$atom$b                                    # Extract the B-factors
  plotb3(x.b, sse=x.chain, typ="l", ylab="Bfactor")        # Plot the B-factors
}
```

We can use PDBID('4AKE'), chainID ('A') as an example.

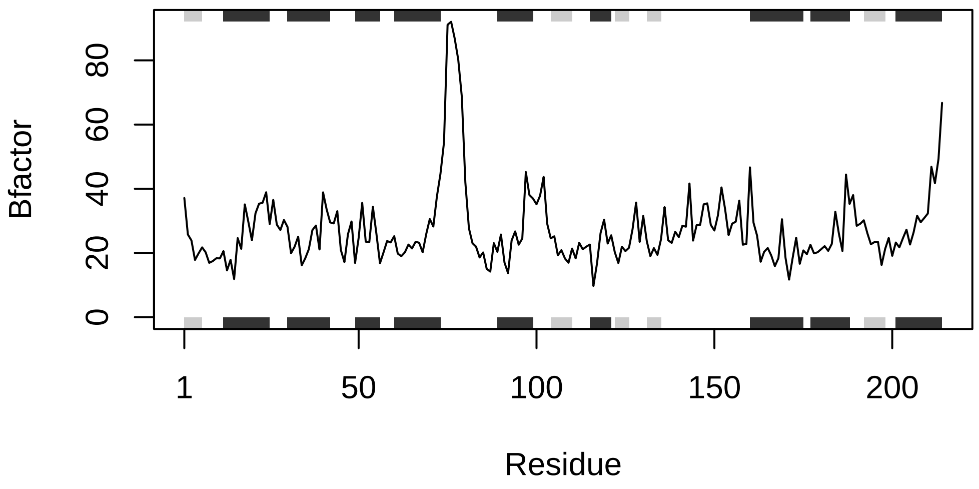
```r
Bfactor("4AKE", "A")
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\18695\AppData\Local\Temp\RtmpiEzw1x/4AKE.pdb exists. Skipping download
```



```r
Bfactor('1AKE', 'A')
```

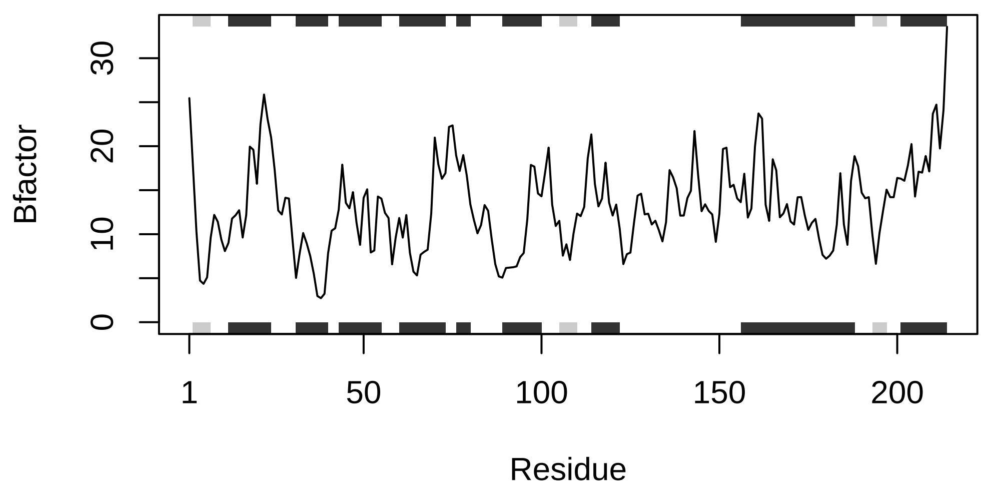
```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\18695\AppData\Local\Temp\RtmpiEzw1x/1AKE.pdb exists. Skipping download

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



```
Bfactor('1E4Y', 'A')
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\18695\AppData\Local\Temp\RtmpiEzw1x/1E4Y.pdb exists. Skipping download
```


Bfactor
0
10
20
30
1
50
100
150
200
Residue