Class 6: HW R Functions

Angela Bartolo PID: A15932451

Section 1: Improving analysis code by writing functions

Part A: Can you improve this analysis 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$a))
df$b <- (df$b - min(df$a)) / (max(df$b) - min(df$b))
df$c <- (df$c - min(df$c)) / (max(df$c) - min(df$c))
df$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
```

Seeing 'df' before and after the script shows that 'df' is transformed with the equations because it shows different numbers than the input data.frame. Next I will remove the common elements to simplify.

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

I will then insert it into a function.

```
df <- data.frame(a=1:10, b=seq(200,400,length=10),c=11:20,d=NA)
parta <- function(x) {
   x <- (x - min(x)) / (max(x) - min(x))
  return(x)
}
parta(df)
```

```
    a  b  c  d
1  NA NA NA NA
2  NA NA NA NA
3  NA NA NA NA
4  NA NA NA NA
5  NA NA NA NA
6  NA NA NA NA
7  NA NA NA NA
8  NA NA NA NA
9  NA NA NA NA
10 NA NA NA NA
```

The above function doesn't work because it takes the min and max of the whole matrix and not each column. So I will insert a for loop to work on each column.

```
df <- data.frame(a=1:10, b=seq(200,400,length=10),c=11:20,d=NA)
parta <- function(x) {
  for(i in 1:ncol(x)) {
   x[,i] <- (x[,i] - min(x[,i])) / (max(x[,i]) - min(x[,i]))
  }
  return(x)
}
parta(df)
```

```
           a         b         c  d
1  0.0000000 0.0000000 0.0000000 NA
2  0.1111111 0.1111111 0.1111111 NA
3  0.2222222 0.2222222 0.2222222 NA
4  0.3333333 0.3333333 0.3333333 NA
5  0.4444444 0.4444444 0.4444444 NA
6  0.5555556 0.5555556 0.5555556 NA
7  0.6666667 0.6666667 0.6666667 NA
8  0.7777778 0.7777778 0.7777778 NA
9  0.8888889 0.8888889 0.8888889 NA
10 1.0000000 1.0000000 1.0000000 NA
```

It worked! :)

Part B: Can you improve this analysis code?

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

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

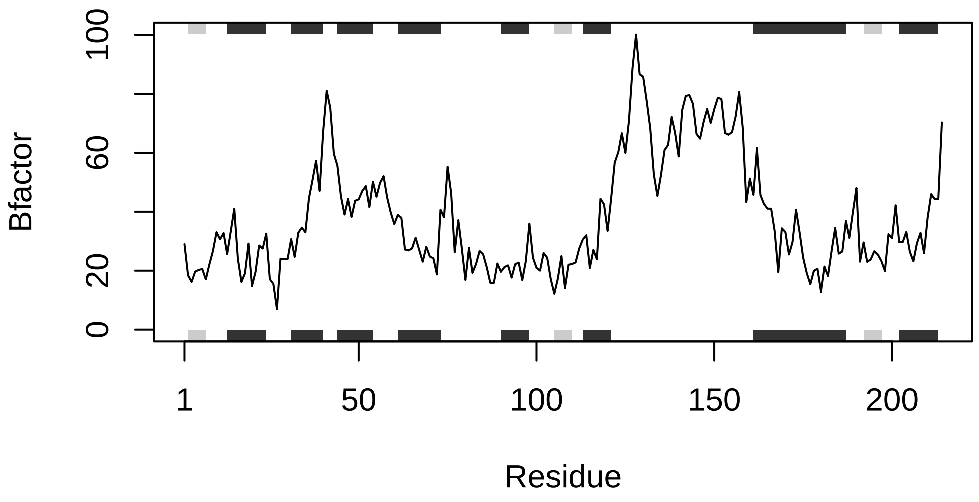
```
s2 <- read.pdb("1AKE") # kinase no drug
```

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

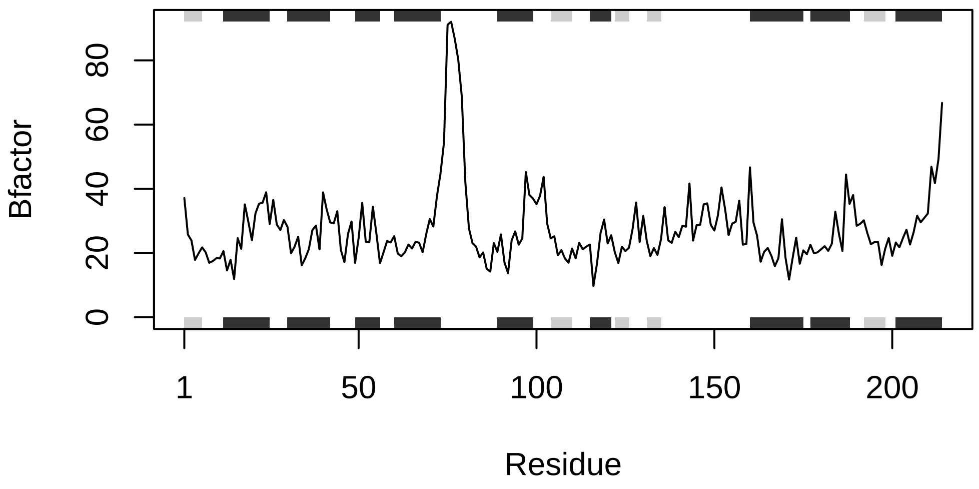
```
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

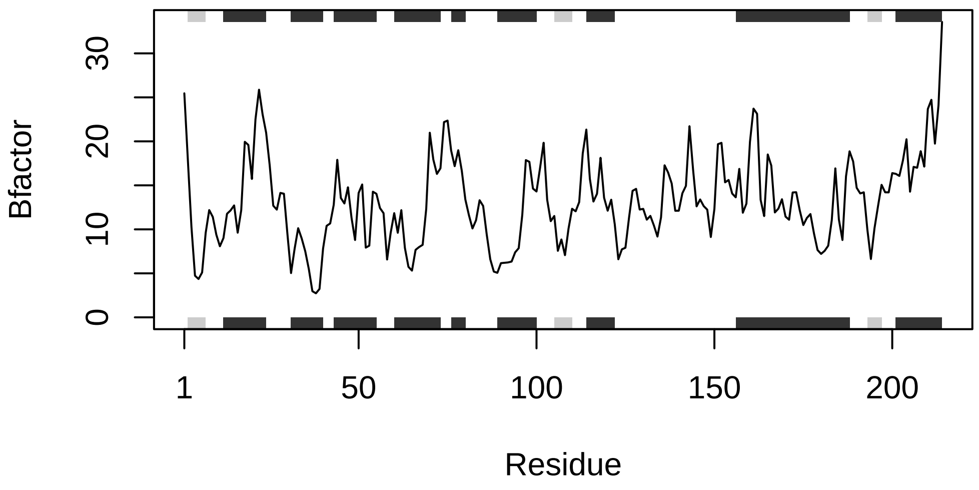
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



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



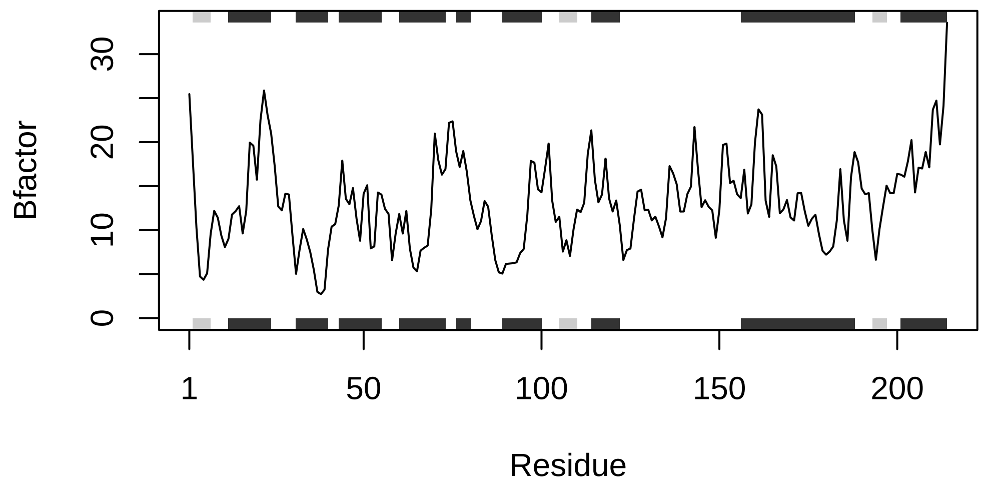
```
plotb3(s3.b, sse=s3.chainA, typ="l", ylab="Bfactor")
```



After fixing the errors, the code will be simplified. And insert into a function.

```r
partb <- function(protein) {
  seq <- read.pdb(protein)

  seq.chainA <- trim.pdb(seq, chain="A", elety="CA")

  seq.b <- seq.chainA$atom$b

  plotb3(seq.b, sse=seq.chainA, typ="l", ylab="Bfactor")
}

partb("1E4Y")
```

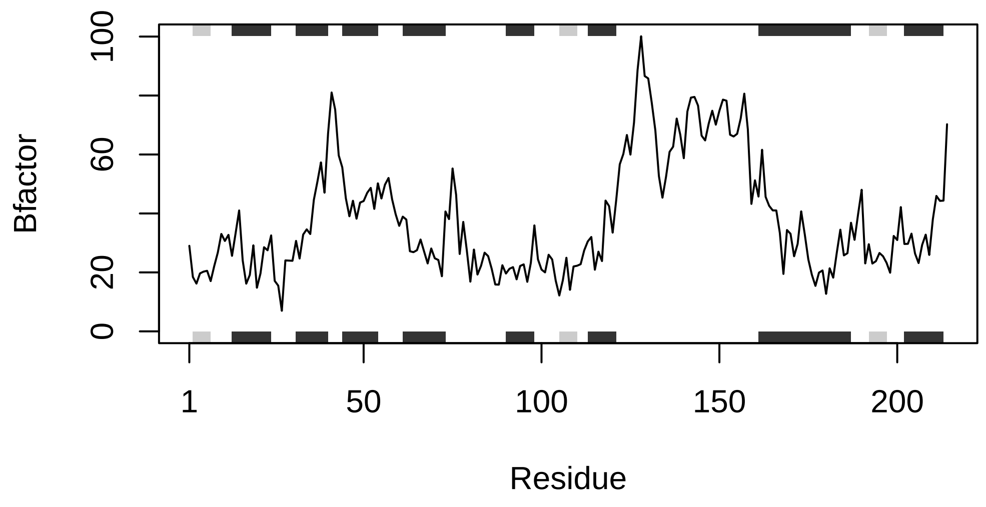
```
  Note: Accessing on-line PDB file

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



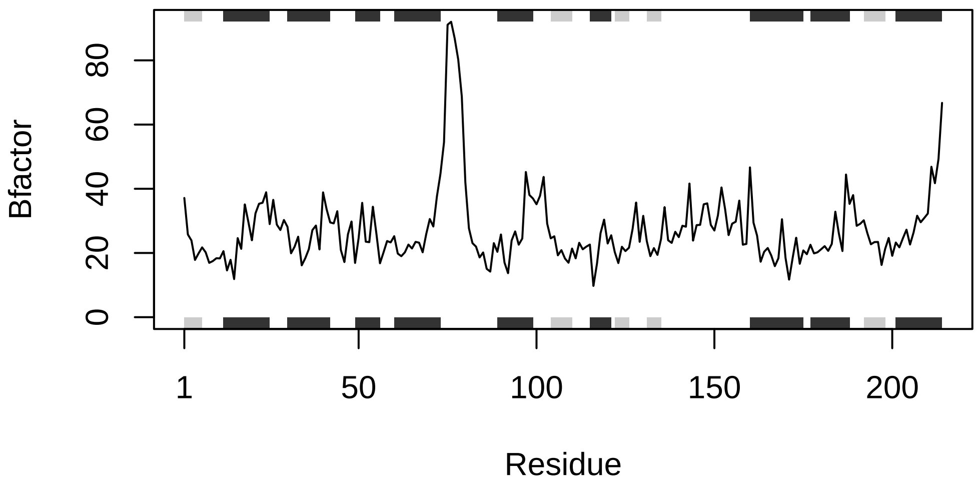
I will add a loop to graph each protein.

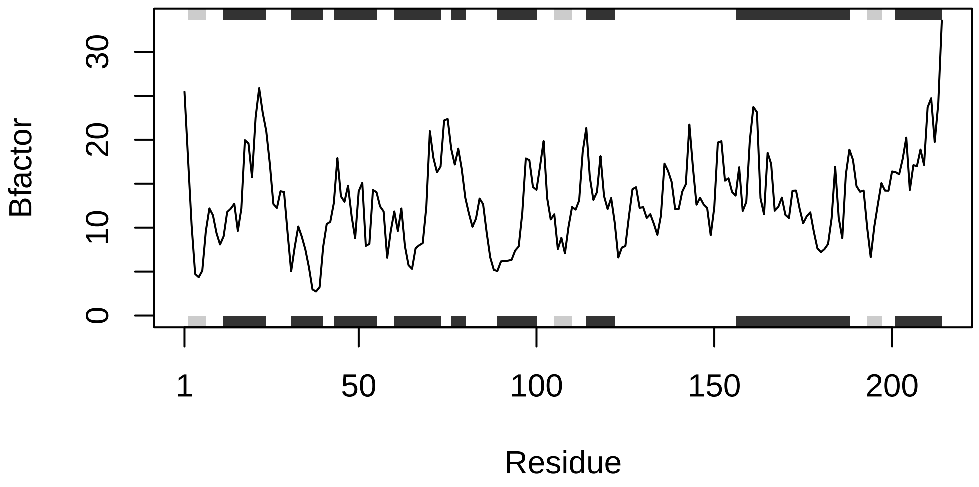
```
proteins <- c("4AKE","1AKE","1E4Y")
partb <- function(proteins) {
  for(i in 1:length(proteins)) {
    seq <- read.pdb(proteins[i])

    seq.chainA <- trim.pdb(seq, chain="A", elety="CA")

    seq.b <- seq.chainA$atom$b

    plotb3(seq.b, sse=seq.chainA, typ="l", ylab="Bfactor")
  }
}

partb(proteins)
```

```
  Note: Accessing on-line PDB file

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

  Note: Accessing on-line PDB file

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



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


  Note: Accessing on-line PDB file


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





Q1. What type of object is returned from the read.pdb() function?

```
#read.pdb("4AKE")
```

"read.pdb()" results in the file of the inputed protein and provides all the data for the protein structure.

> Q2. What does the trim.pdb() function do?

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

"trim.pdb()" cuts part of the protein to only look at a specific section.

> Q3. What input parameter would turn off the marginal black and grey rectangles in the plots and what do they represent in this case?

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

The "sse" is responsible for the marginal black and grey rectangles. They represent the secondary structure of the protein.

> Q4. What would be a better plot to compare across the different proteins?

I think a PCA would work better to compare the different proteins because we could compare the residues and the secondary structure. The secondary structure would be easier to compare since the rectangles do not reflect the secondary structure clearly in these plots.

> Q5. Which proteins are more similar to each other in their B-factor trends. How could you quantify this?

The second and third protein "1AKE" and "1E4Y" have similar B-factor tends since they seems to have a relatively horizontal trend except for the peak in "1AKE". Using the code below quantifies the data.

```
#hc <- hclust( dist( rbind(s1.b, s2.b, s3.b) ) )
#plot(hc)
```

This code puts the elements together into a matrix using 'rbind()'. Then it calculates the distance between the points using 'dist()'. 'hclust()' uses hierarchical clustering to compare the distance by analyzing the dissimilarities and then puts it into a tree.

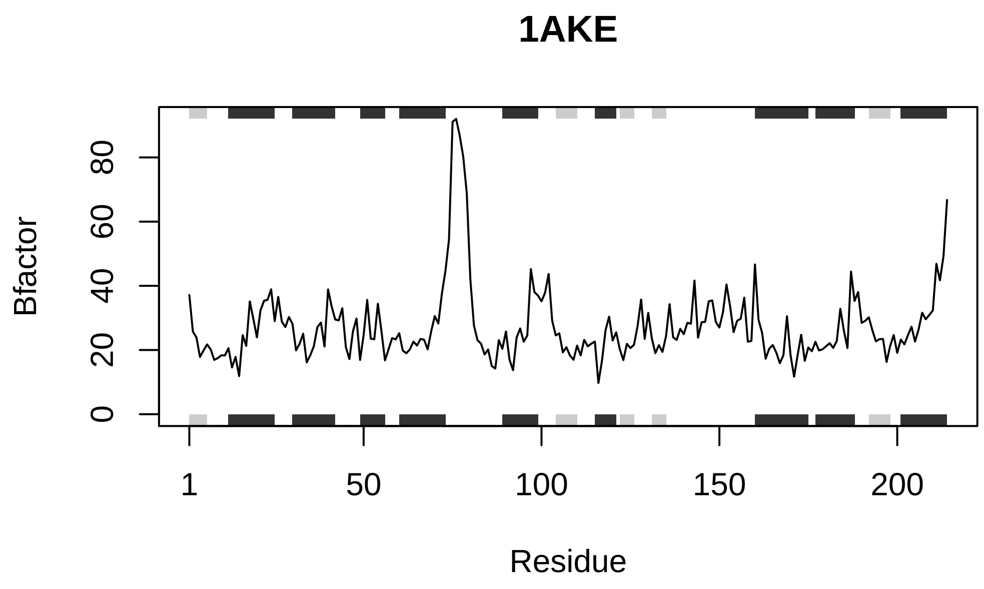
Q6. How would you generalize the original code above to work with any set of input protein structures?

```
partb <- function(...) {
  # The (...) in the function will allow any amount of augments(proteins)

  proteins <- c(...)
  # This will convert the input proteins into a vector

  x <- vector("numeric", 0)
  # This is an empty vector to store all the seq.b (atom values)

  for(i in 1:length(proteins)) {
    # The for loop with read each one of the elements in the vector and produce
    #a graph

    seq <- read.pdb(proteins[i])

    seq.chainA <- trim.pdb(seq, chain="A", elety="CA")

    seq.b <- seq.chainA$atom$b

    plotb3(seq.b, sse=seq.chainA, typ="l", ylab="Bfactor", main= proteins[i])
    # Here I added "main" to give each protein graph a title

    #Next I needed to input and store the seq.b data for each protein into a matrix
    #The "if" statement helps store the seq.b data for the first protein into
    #the empty vector
    #The "else" statement binds the sequential seq.b data for the remaining proteins
    if (i == 1) {
      x <- seq.b
    } else {
        x <- rbind(x, seq.b)
      }
  } #End of "for loop"

  #Add the second line of code from Q5 to compare the proteins
  # I added an if statement because the hc plot would not appear with two
  #proteins and I dont know if it is possible to create an hc plot with
  #only 2 proteins/branches???
  if (length(proteins) >2) {
    rownames(x) <- proteins
    # I added the protein names because it's confusing without them

  hc <- hclust( dist(x))
  plot(hc)
  } else{}

}
```

Test out function with all three proteins and 2 proteins

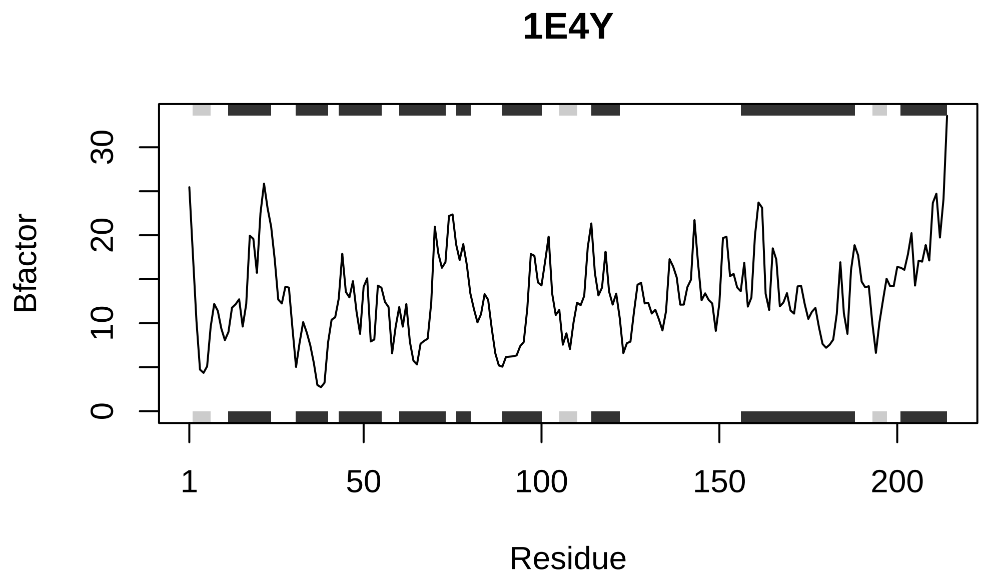
```
partb("1AKE","1E4Y","4AKE")
```

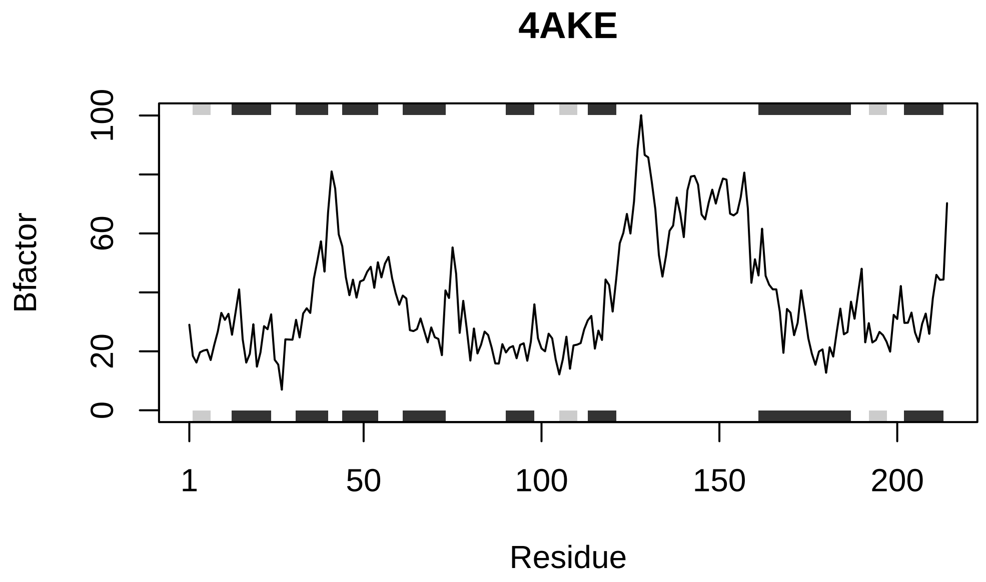
```
  Note: Accessing on-line PDB file

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

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

  Note: Accessing on-line PDB file

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

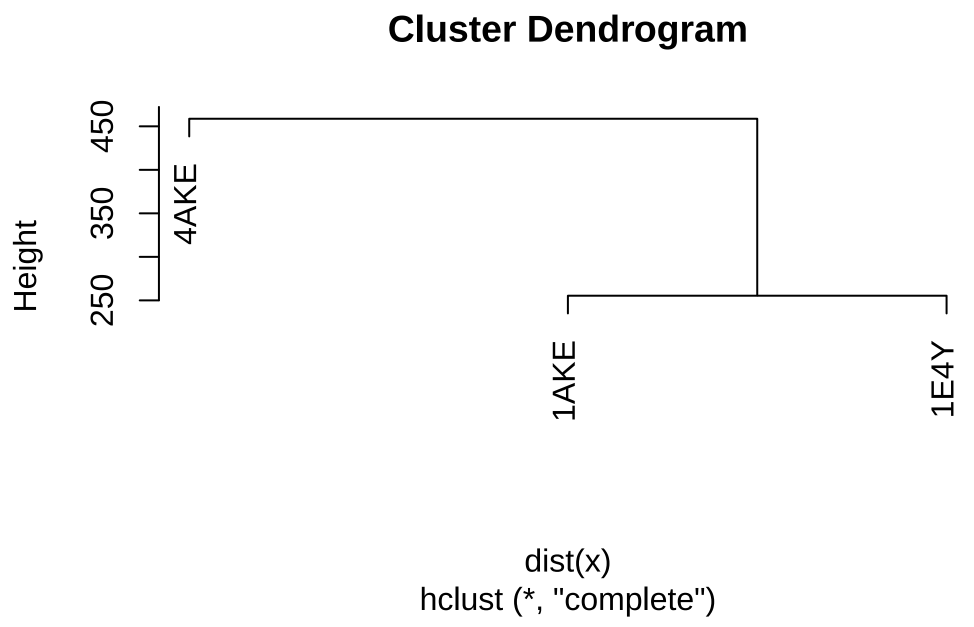


```
  Note: Accessing on-line PDB file
```

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

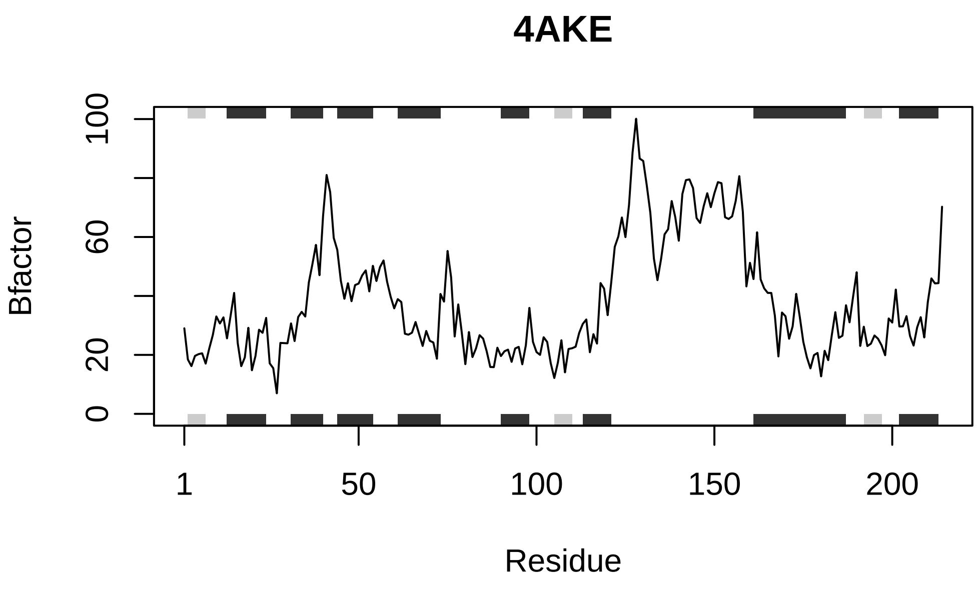

1E4Y
Bfactor
0
10
20
30
1
50
100
150
200
Residue


4AKE
Bfactor
0
20
60
100
1
50
100
150
200
Residue

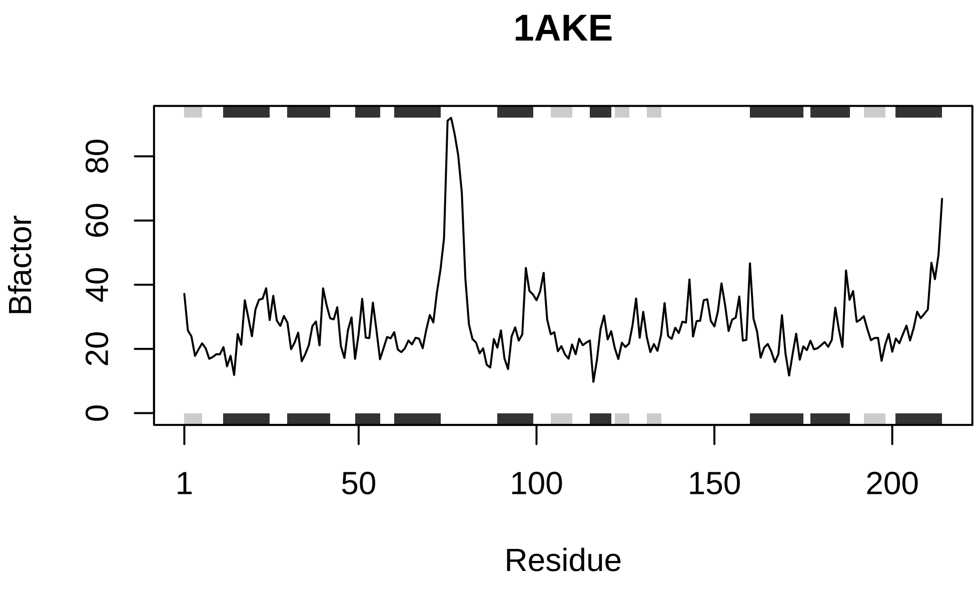


```
partb("4AKE","1AKE")
```

```
  Note: Accessing on-line PDB file

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

  Note: Accessing on-line PDB file

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



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



```
NULL
```