Lab 6: R Functions

Vivian Cai

2/6/2022

Improving analysis code by writing functions

A.

1) Original code:

```
# (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$a <- (df$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))
```

2) Fix copy/paste errors:

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

3) Write a function to clean it up:

```
df <- data.frame(a=1:10, b=seq(200,400,length=10),c=11:20,d=NA)
scaling <- function(x, na.rm = TRUE) {
  rng <- range(x, na.rm = na.rm)
  (x - rng[1]) / (rng[2] - rng[1])
}

scaling(df$a)
```

```
##  [1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444 0.5555556 0.6666667
##  [8] 0.7777778 0.8888889 1.0000000
```

B.

1) Original code:

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

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

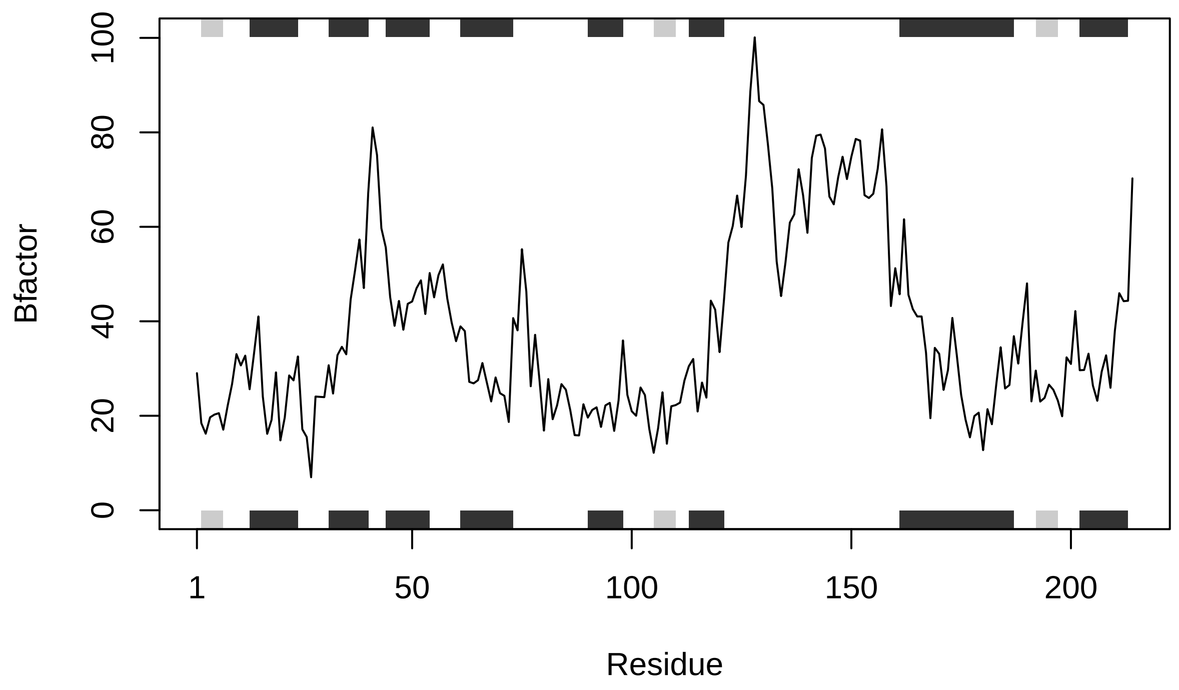
```
s2 <- read.pdb("1AKE")  # kinase no drug
```

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

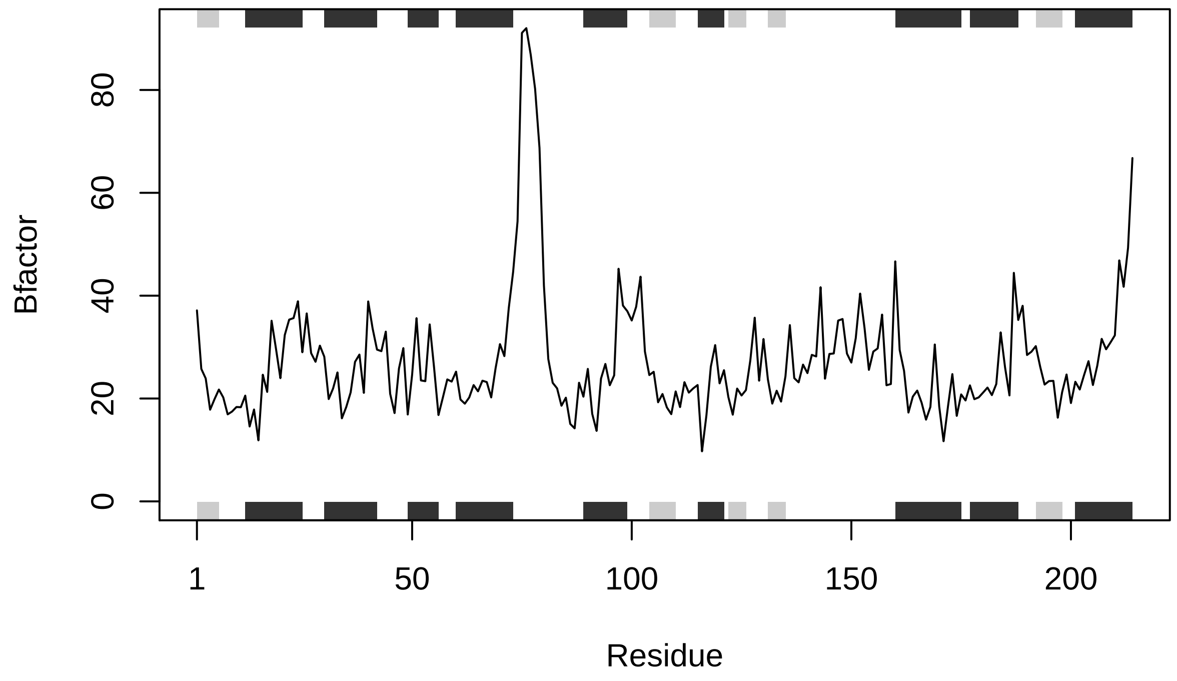
```
s3 <- read.pdb("1E4Y")  # kinase with drug
```

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

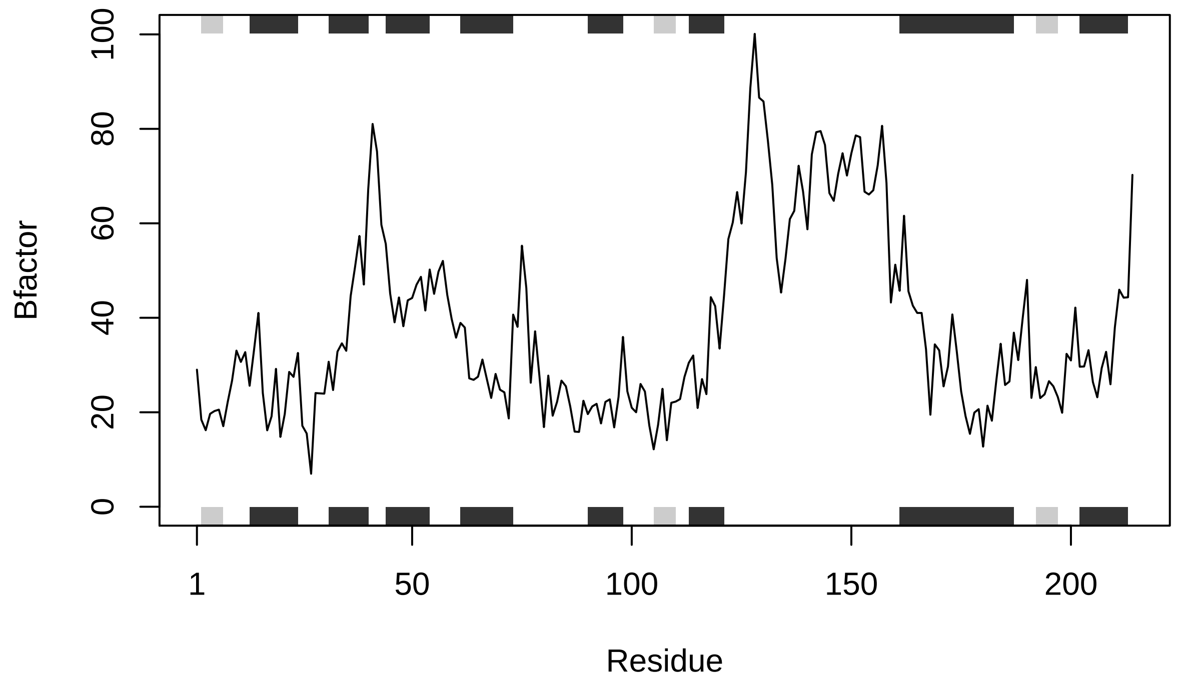
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



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



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



2) Copy/Paste error gone:

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

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/8g/
## c7740b013vd3vszd57ym1z3r0000gn/T//Rtmpz8nUcu/4AKE.pdb exists. Skipping download
```

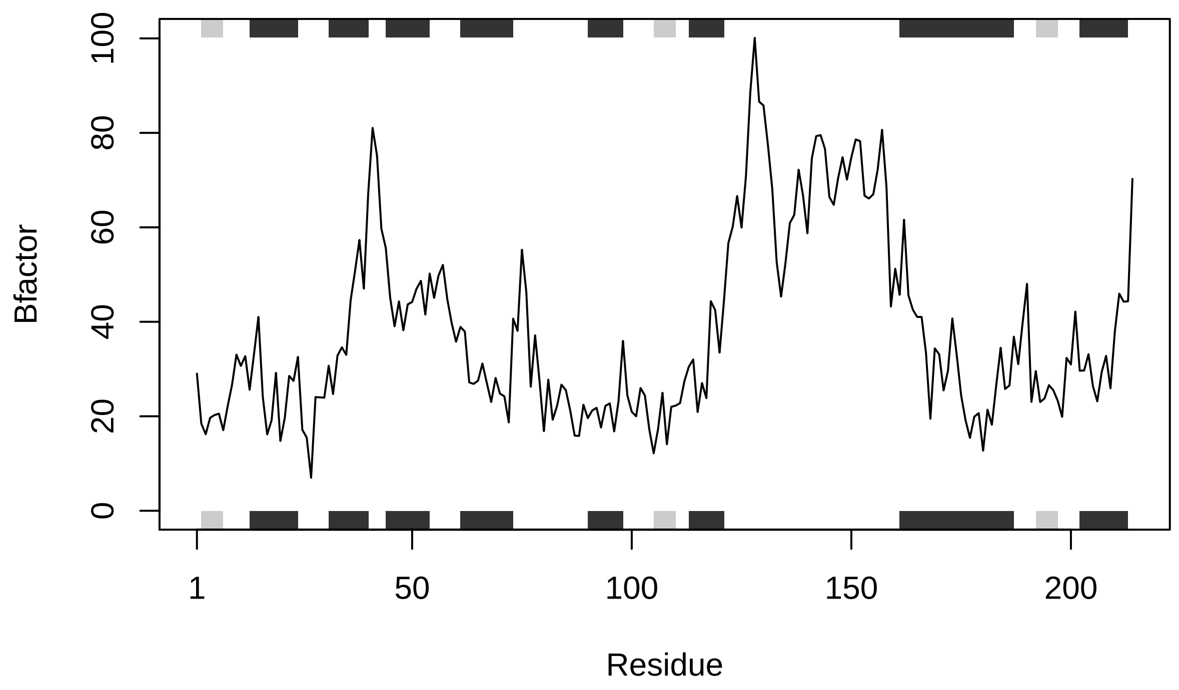
```
s2 <- read.pdb("1AKE")  # kinase no drug
```

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/8g/
## c7740b013vd3vszd57ym1z3r0000gn/T//Rtmpz8nUcu/1AKE.pdb exists. Skipping download

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

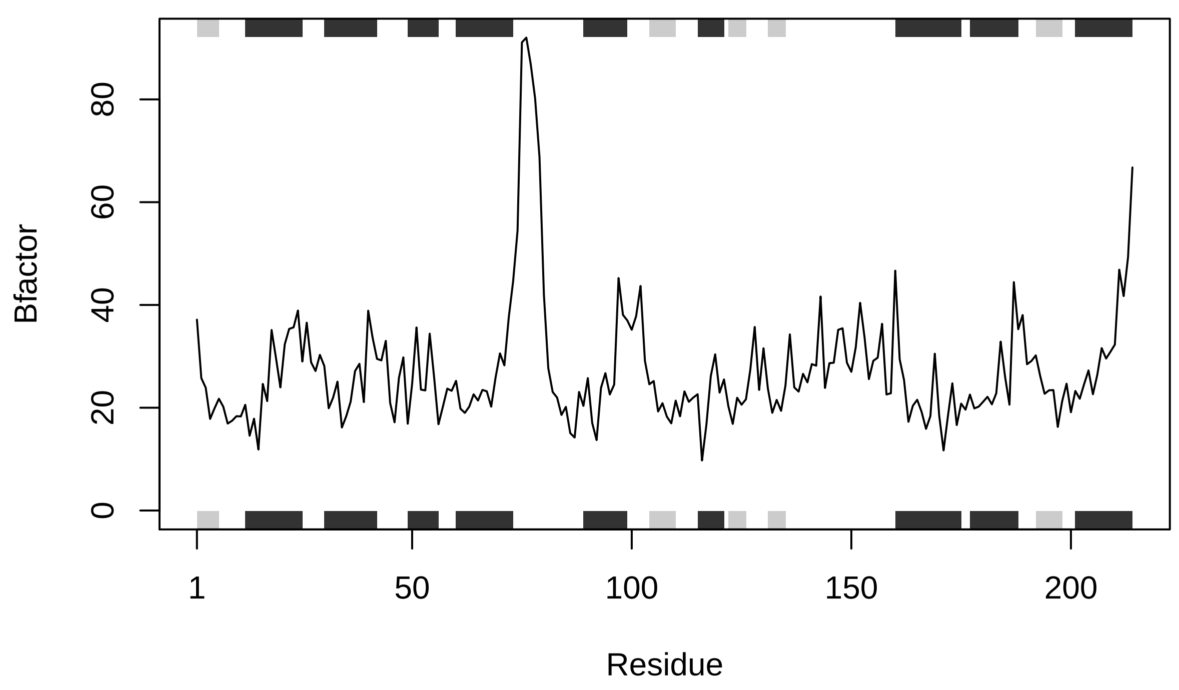
```
s3 <- read.pdb("1E4Y")  # kinase with drug
```

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

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/8g/
## c7740b013vd3vszd57ym1z3r0000gn/T//Rtmpz8nUcu/1E4Y.pdb exists. Skipping download
```

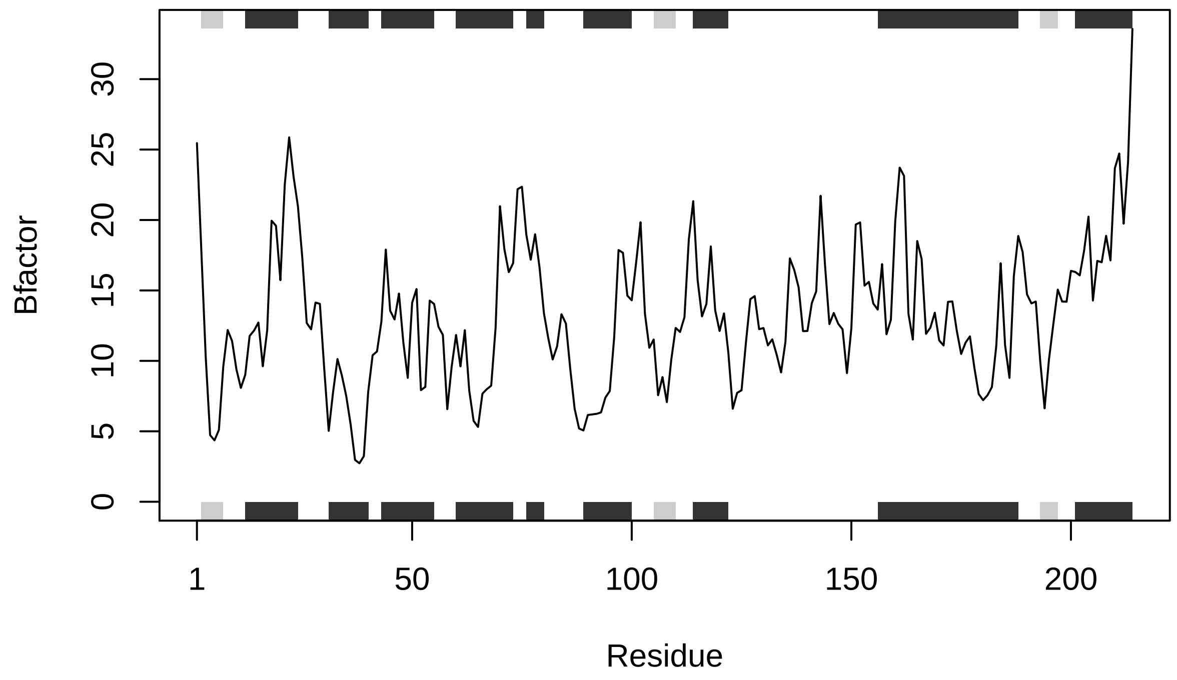
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



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



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

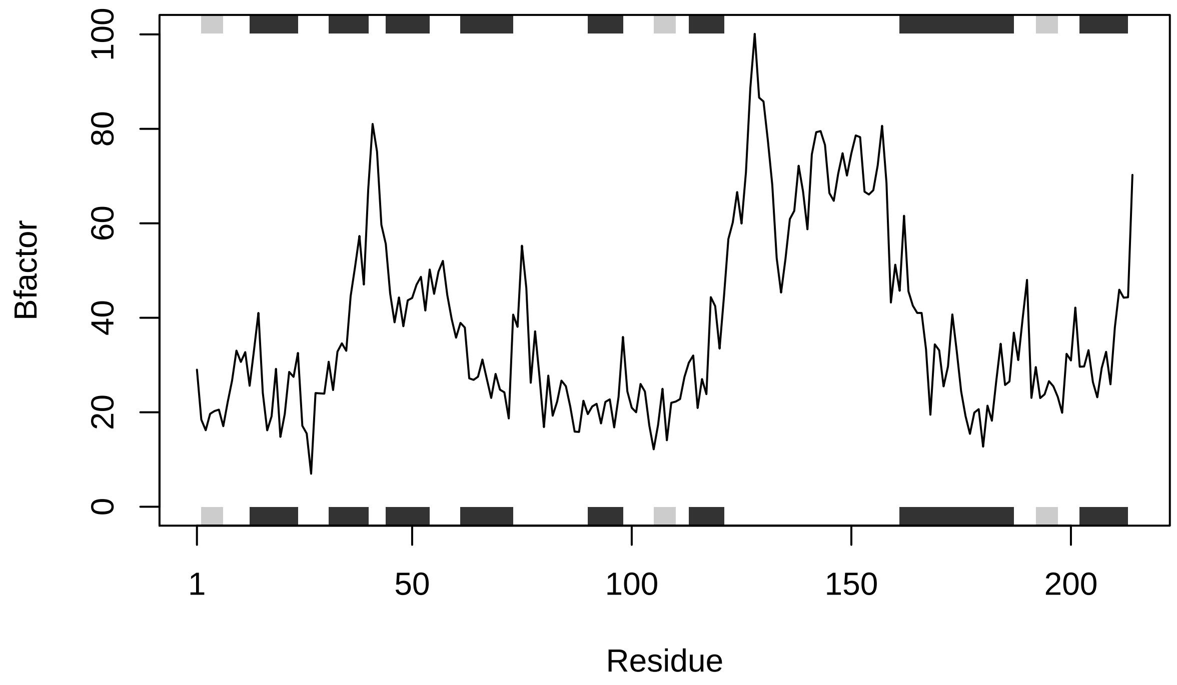


3) Clean up with a new function:

```
PDB_Bfac_plt <- function(x, chain = "A", elety = "CA", typ = "l", ylab = "Bfactor") {
  PDB <- read.pdb(x)
  Chain_PDB <- trim.pdb(PDB, chain = chain, elety = elety)
  Chain_atomb <- Chain_PDB$atom$b
  plotb3(Chain_atomb, sse = Chain_PDB, typ = typ, ylab = ylab)
}

PDB_Bfac_plt("4AKE")
```

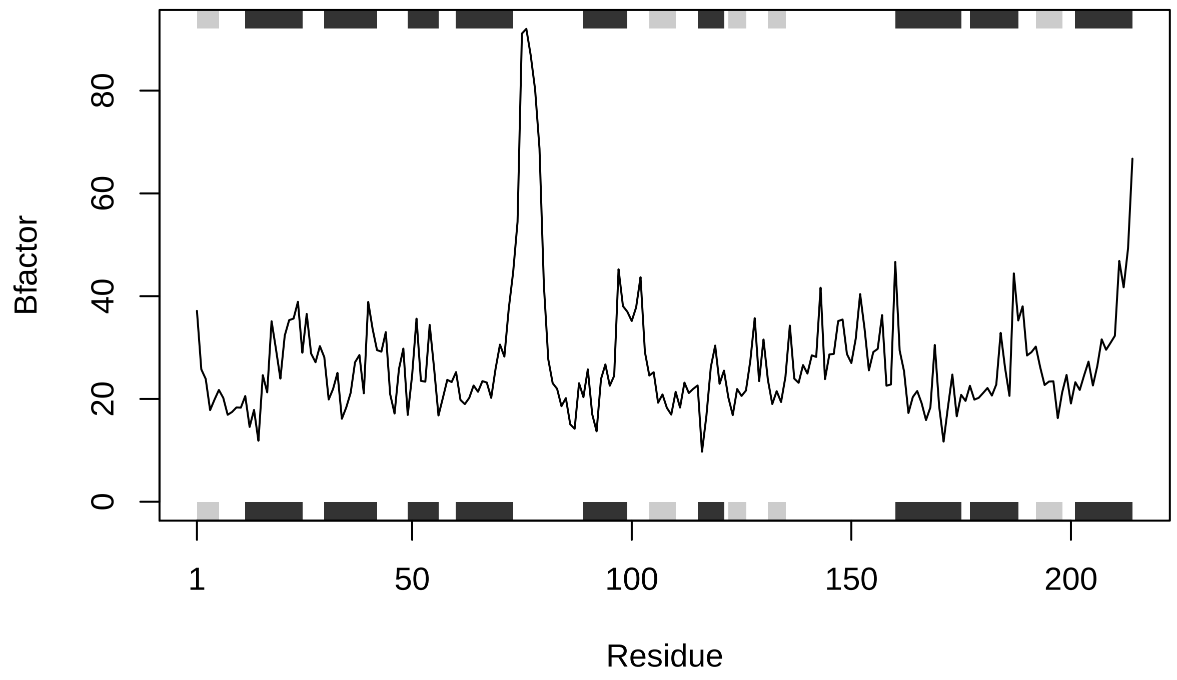
```
##   Note: Accessing on-line PDB file
```

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/8g/
## c7740b013vd3vszd57ym1z3r0000gn/T//Rtmpz8nUcu/4AKE.pdb exists. Skipping download
```



```
PDB_Bfac_plt("1AKE")
```

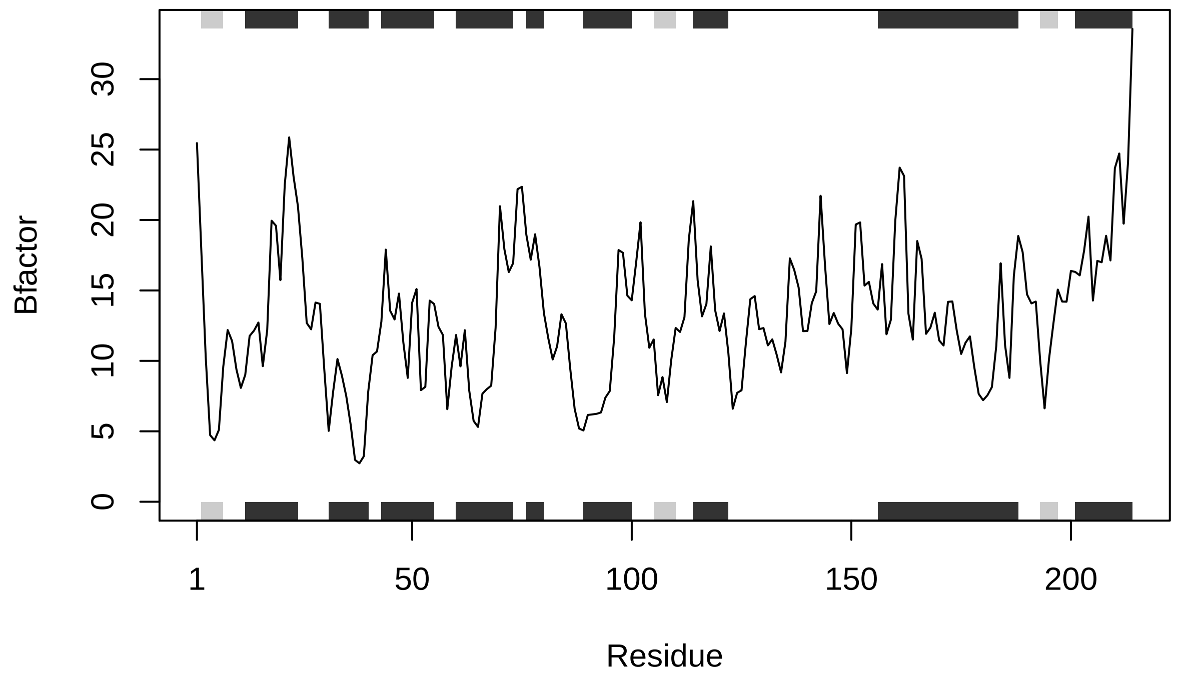
```
##   Note: Accessing on-line PDB file

## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/8g/
## c7740b013vd3vszd57ym1z3r0000gn/T//Rtmpz8nUcu/1AKE.pdb exists. Skipping download

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



```
PDB_Bfac_plt("1E4Y")
```

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

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/8g/
## c7740b013vd3vszd57ym1z3r0000gn/T//Rtmpz8nUcu/1E4Y.pdb exists. Skipping download
```



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

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

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

```
## Warning in get.pdb(file, path = tempdir(), verbose = FALSE): /var/folders/8g/
## c7740b013vd3vszd57ym1z3r0000gn/T//Rtmpz8nUcu/4AKE.pdb exists. Skipping download
```

```
## List of 8
##  $ atom  :'data.frame':  3459 obs. of  16 variables:
##   ..$ type  : chr [1:3459] "ATOM" "ATOM" "ATOM" "ATOM" ...
##   ..$ eleno : int [1:3459] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ elety : chr [1:3459] "N" "CA" "C" "O" ...
##   ..$ alt   : chr [1:3459] NA NA NA NA ...
##   ..$ resid : chr [1:3459] "MET" "MET" "MET" "MET" ...
##   ..$ chain : chr [1:3459] "A" "A" "A" "A" ...
##   ..$ resno : int [1:3459] 1 1 1 1 1 1 1 1 2 2 ...
##   ..$ insert: chr [1:3459] NA NA NA NA ...
##   ..$ x     : num [1:3459] -10.93 -9.9 -9.17 -9.8 -10.59 ...
##   ..$ y     : num [1:3459] -24.9 -24.4 -23.3 -22.3 -24 ...
##   ..$ z     : num [1:3459] -9.52 -10.48 -9.81 -9.35 -11.77 ...
##   ..$ o     : num [1:3459] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ b     : num [1:3459] 41.5 29 27.9 26.4 34.2 ...
##   ..$ segid : chr [1:3459] NA NA NA NA ...
##   ..$ elesy : chr [1:3459] "N" "C" "C" "O" ...
##   ..$ charge: chr [1:3459] NA NA NA NA ...
##  $ xyz   : 'xyz' num [1, 1:10377] -10.93 -24.89 -9.52 -9.9 -24.42 ...
##  $ seqres: Named chr [1:428] "MET" "ARG" "ILE" "ILE" ...
##   ..- attr(*, "names")= chr [1:428] "A" "A" "A" "A" ...
##  $ helix :List of 4
##   ..$ start: Named num [1:19] 13 31 44 61 75 90 113 161 202 13 ...
##   .. ..- attr(*, "names")= chr [1:19] "" "" "" "" ...
##   ..$ end  : Named num [1:19] 24 40 54 73 77 98 121 187 213 24 ...
##   .. ..- attr(*, "names")= chr [1:19] "" "" "" "" ...
##   ..$ chain: chr [1:19] "A" "A" "A" "A" ...
##   ..$ type : chr [1:19] "5" "1" "1" "1" ...
##  $ sheet :List of 4
##   ..$ start: Named num [1:14] 192 105 2 81 27 123 131 192 105 2 ...
##   .. ..- attr(*, "names")= chr [1:14] "" "" "" "" ...
##   ..$ end  : Named num [1:14] 197 110 7 84 29 126 134 197 110 7 ...
##   .. ..- attr(*, "names")= chr [1:14] "" "" "" "" ...
##   ..$ chain: chr [1:14] "A" "A" "A" "A" ...
##   ..$ sense: chr [1:14] "0" "1" "1" "1" ...
##  $ calpha: logi [1:3459] FALSE TRUE FALSE FALSE FALSE FALSE ...
##  $ remark:List of 1
##   ..$ biomat:List of 4
##   .. ..$ num   : int 1
##   .. ..$ chain :List of 1
##   .. .. ..$ : chr [1:2] "A" "B"
##   .. ..$ mat   :List of 1
##   .. .. ..$ :List of 1
##   .. .. .. ..$ A B: num [1:3, 1:4] 1 0 0 0 1 0 0 0 1 0 ...
##   .. ..$ method: chr "AUTHOR"
##  $ call  : language read.pdb(file = "4AKE")
##  - attr(*, "class")= chr [1:2] "pdb" "sse"
```

A data frame is returned.

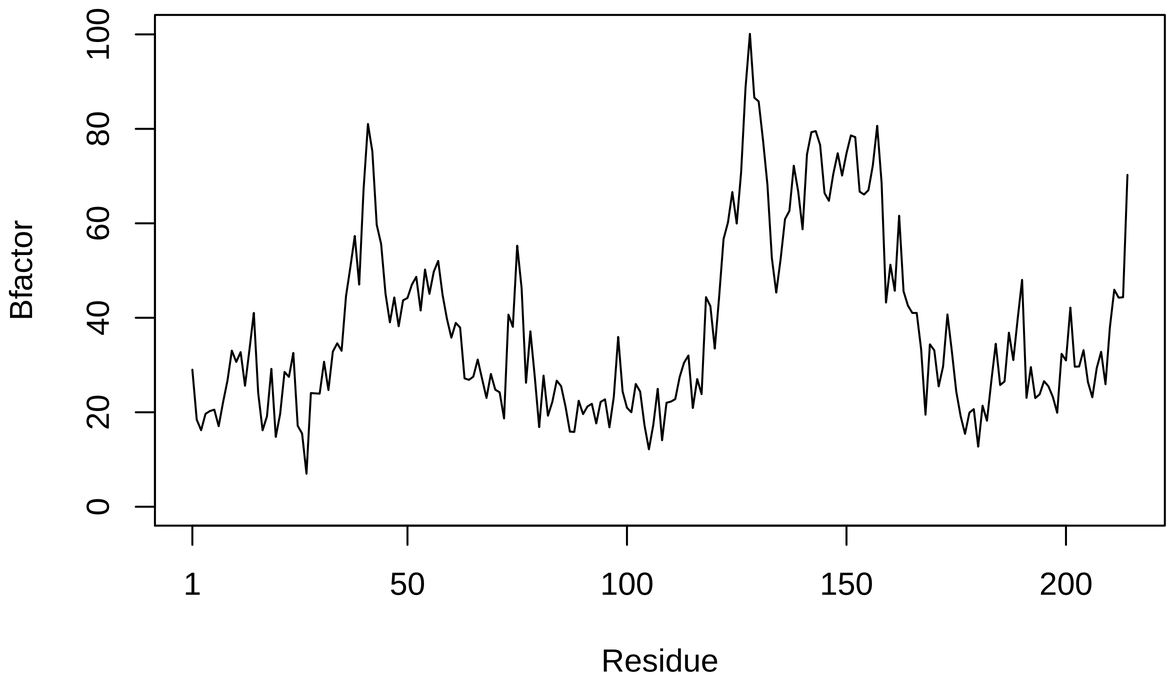
Q2. What does the trim.pdb() function do?

```
?trim.pdb
```

It creates a smaller PDB from a given PDB object, containing a subset of atoms from the original PDB.

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=NULL, typ="l", ylab="Bfactor")
```

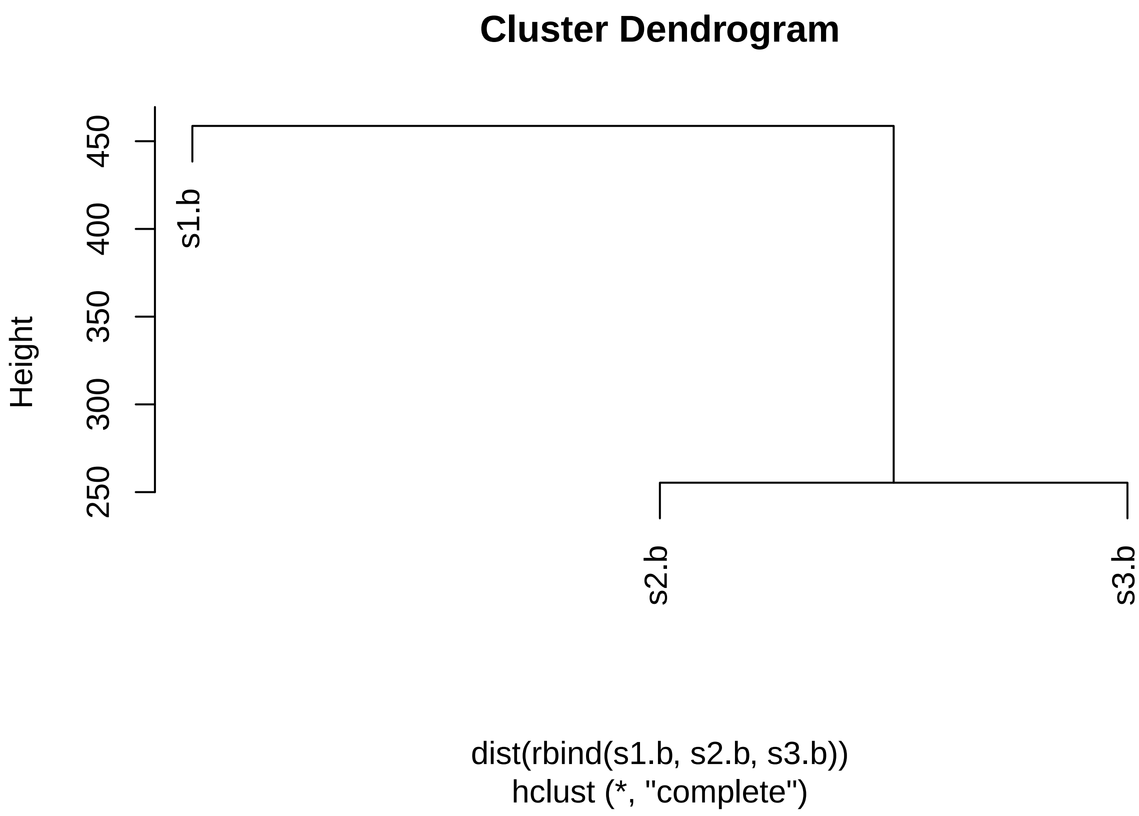


As shown, turning off the sse argument gets rid of the bars, which are indications of secondary structures as returned from dssp, stride or in certsin cases read.pdb.

Q4. What would be a better plot to compare across the different proteins? I think the same line plot with 3 Bfactor lines showing on the same plit would be good for comparison, since all 3 proteins of interest are the same length.

Q5. Which proteins are more similar to each other in their B-factor trends. How could you quantify this? HINT: try the rbind(), dist() and hclust() functions together with a resulting dendrogram plot. Look up the documentation to see what each of these functions does.

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



rbind combines the rows of a chosen data.frame, in this case, all the b-factor values for each protein we are looking at are combined into the same dataframe.(As shown below)

```
head(rbind(s1.b, s2.b, s3.b))
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## s1.b 29.02 18.44 16.20 19.67 20.26 20.55 17.05 22.13 26.71 33.05 30.66 32.73
## s2.b 37.14 25.76 23.90 17.83 19.86 21.75 20.21 16.92 17.47 18.35 18.31 20.57
## s3.b 25.46 17.86 10.28  4.73  4.36  5.10  9.59 12.19 11.41  9.39  8.08  9.01
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## s1.b 25.61 33.19 41.03 24.09 16.18 19.14 29.19 14.79 19.63 28.54 27.49 32.56
## s2.b 14.56 17.87 11.87 24.63 21.29 35.13 29.68 23.96 32.34 35.34 35.64 38.91
## s3.b 11.77 12.15 12.72  9.62 12.18 19.95 19.59 15.73 22.51 25.87 23.08 20.97
##      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36]
## s1.b 17.13 15.50  6.98 24.07 24.00 23.94 30.70 24.70 32.84 34.60 33.01 44.60
## s2.b 29.00 36.55 28.83 27.15 30.28 28.13 19.90 21.95 25.07 16.15 18.35 21.19
## s3.b 17.28 12.69 12.24 14.14 14.05  9.38  5.03  7.78 10.13  8.96  7.50  5.48
##      [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48]
## s1.b 50.74 57.32 47.04 67.13 81.04 75.20 59.68 55.63 45.12 39.04 44.31 38.21
## s2.b 27.13 28.55 21.10 38.88 33.63 29.51 29.21 33.01 20.92 17.17 25.84 29.80
## s3.b  2.97  2.73  3.23  7.81 10.40 10.67 12.79 17.90 13.56 12.94 14.78 11.31
##      [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60]
## s1.b 43.70 44.19 47.00 48.67 41.54 50.22 45.07 49.77 52.04 44.82 39.75 35.79
## s2.b 16.89 24.66 35.62 23.52 23.37 34.41 25.96 16.79 20.20 23.72 23.29 25.23
## s3.b  8.79 14.13 15.10  7.92  8.15 14.28 14.04 12.42 11.84  6.57  9.59 11.84
##      [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72]
## s1.b 38.92 37.93 27.18 26.86 27.53 31.16 27.08 23.03 28.12 24.78 24.22 18.69
## s2.b 19.81 19.00 20.21 22.62 21.40 23.47 23.20 20.21 25.90 30.58 28.25 37.60
## s3.b  9.61 12.18  7.89  5.74  5.31  7.67  7.99  8.24 12.34 20.98 17.93 16.30
##      [,73] [,74] [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84]
## s1.b 40.67 38.08 55.26 46.29 26.25 37.14 27.50 16.86 27.76 19.27 22.22 26.70
## s2.b 44.66 54.46 91.10 92.02 86.85 80.21 68.72 42.01 27.69 23.06 21.98 18.60
## s3.b 16.94 22.19 22.36 18.96 17.18 18.99 16.65 13.39 11.61 10.10 11.03 13.31
##      [,85] [,86] [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96]
## s1.b 25.52 21.22  15.9 15.84 22.44 19.61 21.23 21.79 17.64 22.19 22.73 16.80
## s2.b 20.17 15.06  14.2 23.07 20.36 25.76 17.02 13.71 23.88 26.72 22.58 24.51
## s3.b 12.66  9.44   6.6  5.20  5.06  6.16  6.20  6.24  6.34  7.39  7.86 11.66
##      [,97] [,98] [,99] [,100] [,101] [,102] [,103] [,104] [,105] [,106] [,107]
## s1.b 23.25 35.95 24.42  20.96  20.00  25.99  24.39  17.19  12.16  17.35  24.97
## s2.b 45.23 38.07 36.97  35.17  37.83  43.69  29.14  24.56  25.20  19.27  20.88
## s3.b 17.87 17.67 14.63  14.30  16.98  19.84  13.36  10.93  11.52   7.56   8.85
##      [,108] [,109] [,110] [,111] [,112] [,113] [,114] [,115] [,116] [,117]
## s1.b  14.08  22.01  22.26  22.78  27.47  30.49  32.02  20.90  27.03  23.84
## s2.b  18.27  16.96  21.38  18.33  23.18  21.15  21.97  22.63   9.74  16.71
## s3.b   7.07  10.08  12.34  12.05  13.10  18.63  21.34  15.73  13.16  14.04
##      [,118] [,119] [,120] [,121] [,122] [,123] [,124] [,125] [,126] [,127]
## s1.b  44.37  42.47  33.48  44.56  56.67  60.18  66.62  59.95  70.81  88.63
## s2.b  26.18  30.39  22.95  25.51  20.28  16.86  21.94  20.59  21.64  27.42
## s3.b  18.13  13.59  12.12  13.37  10.57   6.60   7.73   7.91  11.31  14.38
##      [,128] [,129] [,130] [,131] [,132] [,133] [,134] [,135] [,136] [,137]
## s1.b 100.11  86.60  85.80  77.48  68.13  52.66  45.34  52.43  60.90  62.64
## s2.b  35.72  23.47  31.57  23.71  19.01  21.52  19.40  24.32  34.28  23.96
## s3.b  14.60  12.25  12.33  11.10  11.53  10.44   9.18  11.36  17.28  16.45
##      [,138] [,139] [,140] [,141] [,142] [,143] [,144] [,145] [,146] [,147]
## s1.b  72.19  66.75  58.73  74.57  79.29  79.53  76.58  66.40  64.76  70.48
## s2.b  23.14  26.60  24.94  28.49  28.18  41.64  23.85  28.67  28.76  35.16
## s3.b  15.21  12.11  12.12  14.10  14.94  21.72  16.82  12.61  13.40  12.64
##      [,148] [,149] [,150] [,151] [,152] [,153] [,154] [,155] [,156] [,157]
## s1.b  74.84  70.11  74.82  78.61  78.24  66.70  66.10  67.01  72.28  80.64
## s2.b  35.46  28.74  26.99  31.74  40.41  33.73  25.57  29.13  29.74  36.32
## s3.b  12.24   9.13  12.31  19.68  19.83  15.34  15.61  14.07  13.64  16.87
##      [,158] [,159] [,160] [,161] [,162] [,163] [,164] [,165] [,166] [,167]
## s1.b  68.54  43.23  51.24  45.72  61.60  45.61  42.57  41.03  41.02  33.34
## s2.b  22.58  22.82  46.67  29.44  25.40  17.27  20.38  21.55  19.19  15.89
## s3.b  11.89  12.92  19.93  23.72  23.13  13.35  11.51  18.51  17.24  11.92
##      [,168] [,169] [,170] [,171] [,172] [,173] [,174] [,175] [,176] [,177]
## s1.b  19.48  34.38  33.11  25.48  29.68  40.71  32.91  24.41  19.20  15.43
## s2.b  18.37  30.51  18.47  11.70  18.45  24.75  16.63  20.80  19.62  22.56
## s3.b  12.36  13.42  11.45  11.09  14.19  14.22  12.15  10.49  11.29  11.74
##      [,178] [,179] [,180] [,181] [,182] [,183] [,184] [,185] [,186] [,187]
## s1.b  19.93  20.66  12.72  21.40  18.21  26.68  34.50  25.77  26.52  36.85
## s2.b  19.87  20.22  21.16  22.13  20.66  22.82  32.86  26.04  20.60  44.44
## s3.b   9.53   7.65   7.21   7.56   8.14  11.07  16.93  11.12   8.79  16.03
##      [,188] [,189] [,190] [,191] [,192] [,193] [,194] [,195] [,196] [,197]
## s1.b  31.05  39.84  48.03  23.04  29.57  23.00  23.80  26.59  25.49  23.25
## s2.b  35.28  38.03  28.46  29.10  30.19  26.17  22.71  23.39  23.44  16.27
## s3.b  18.87  17.72  14.72  14.08  14.21   9.99   6.63  10.11  12.64  15.06
##      [,198] [,199] [,200] [,201] [,202] [,203] [,204] [,205] [,206] [,207]
## s1.b  19.89  32.37  30.97  42.16  29.64  29.69  33.15  26.38  23.17  29.35
## s2.b  21.26  24.67  19.12  23.26  21.75  24.59  27.26  22.63  26.40  31.60
## s3.b  14.21  14.20  16.39  16.31  16.07  17.83  20.24  14.28  17.10  17.00
##      [,208] [,209] [,210] [,211] [,212] [,213] [,214]
## s1.b  32.80  25.92  38.01  45.95  44.26  44.35  70.26
## s2.b  29.57  30.90  32.29  46.86  41.73  49.31  66.76
## s3.b  18.88  17.13  23.68  24.72  19.74  24.12  33.57
```

Then, dist computes the distance between rows of the gicen data.frame. Therefore, each element in each row is considered and compared with the other rows. The dist result caculated is then used to make the dendrogram. In our case, s1 and s3 are more similar to each other than they are to s2.

Q6. How would you generalize the original code above to work with any set of input protein structures? This is already done using my PDB_Bfac_plt() function.