HW6 - Lab supplement

Gaeun Jun: A16814573

```r
library(bio3d)

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

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

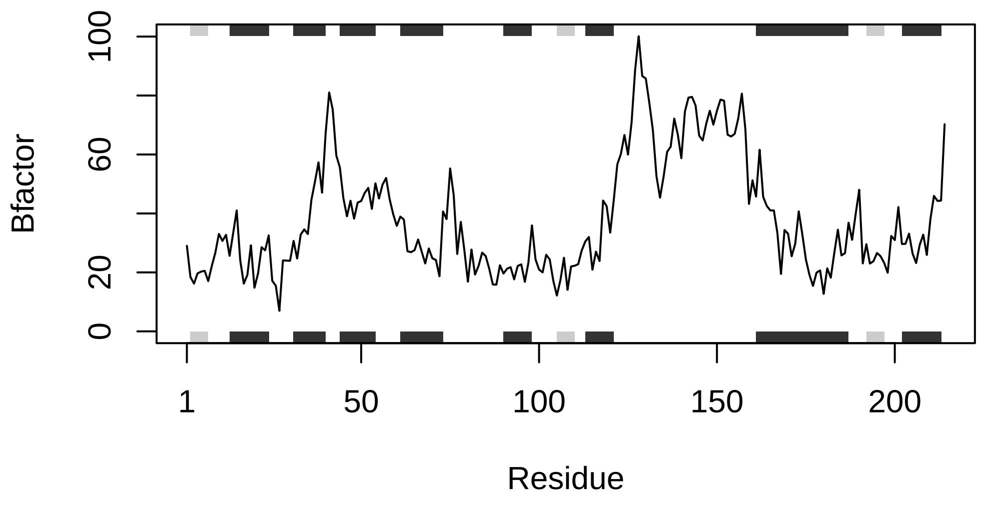
```r
s2 <- read.pdb("1AKE") # kinase no drug
```

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

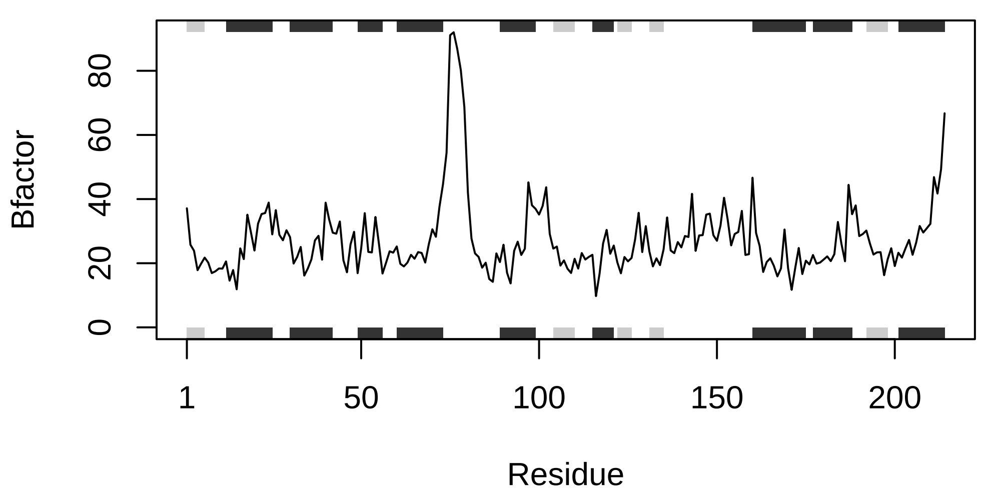
```r
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

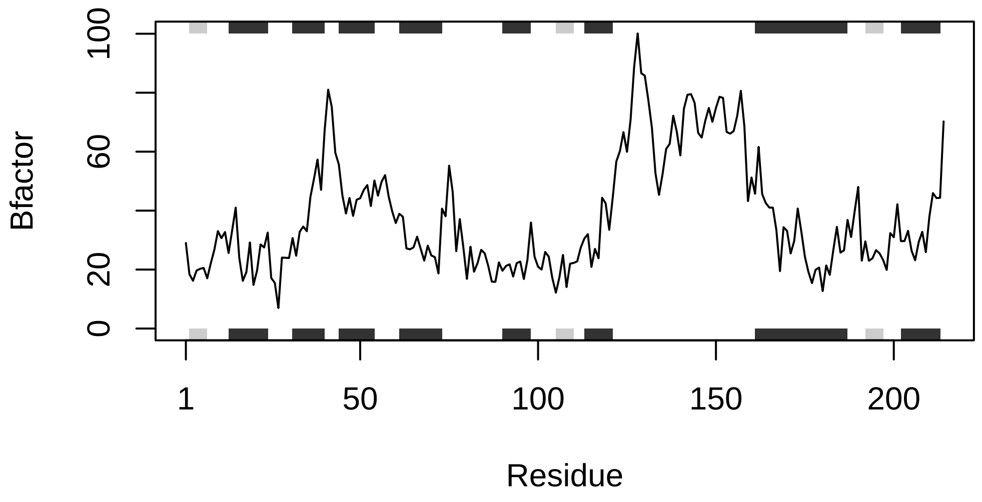
```r
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



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



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



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

```r
str(s1)
```

```
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 list is returned from the read.pdb() function.

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

```r
s1.chainA
```

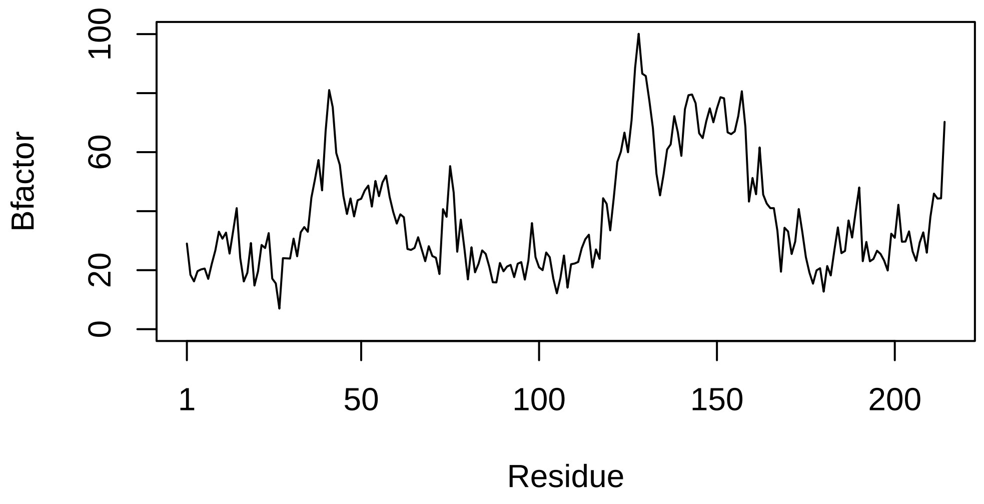
```

 Call:  trim.pdb(pdb = s1, chain = "A", elety = "CA")

   Total Models#: 1
   Total Atoms#: 214,  XYZs#: 642  Chains#: 1  (values: A)

   Protein Atoms#: 214  (residues/Calpha atoms#: 214)
   Nucleic acid Atoms#: 0  (residues/phosphate atoms#: 0)

   Non-protein/nucleic Atoms#: 0  (residues: 0)
   Non-protein/nucleic resid values: [ none ]

  Protein sequence:
     MRIILLGAPGAGKGTQAQFIMEKYGIPQISTGDMLRAAVKSGSELGKQAKDIMDAGKLVT
     DELVIALVKERIAQEDCRNGFLLDGFPRTIPQADAMKEAGINVDYVLEFDVPDELIVDRI
     VGRRVHAPSGRVYHVKFNPPKVEGKDDVTGEELTTRKDDQEETVRKRLVEYHQMTAPLIG
     YYSKEAEAGNTKYAKVDGTKPVAEVRADLEKILG

+ attr: atom, helix, sheet, seqres, xyz,
        calpha, call
```

The trim.pdb() function allows me to call a specific model type in the list of atom data frames. For the specified findings, like `chain` and `elety`, using the trim function can find the specified chain and element type of the atom list, essentially "trimming" down the many options that could be found within the data frame.

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

```r
?plotb3
```

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



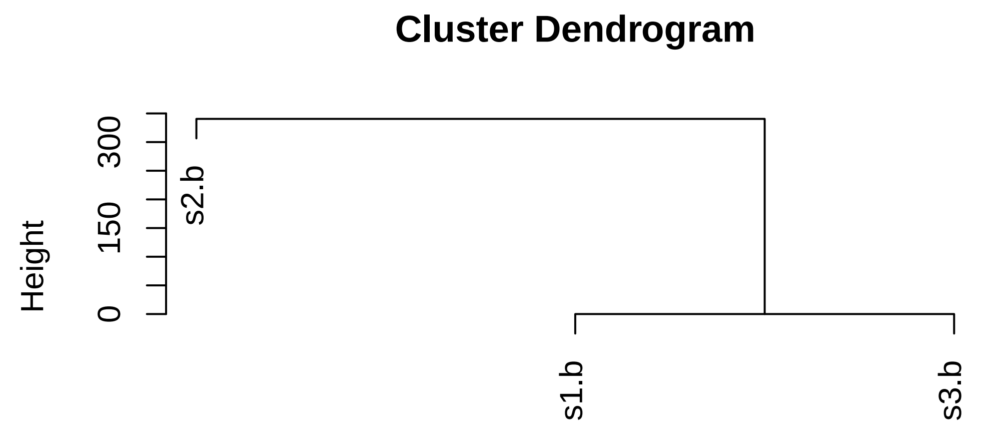
The parameters `top` and `bottom` = FALSE would turn the marginal black and grey rectangles off. They indicate the range of the residue values.

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

A better plot to compare across the different proteins would probably be one that can merge the plots together (while still maintaining a line plot), with different colors to represent the different line plots. A good legend with the colors and representations would be needed as well.

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

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



dist(rbind(s1.b, s2.b, s3.b))
hclust (*, "complete")

The s1.b and s3.b proteins are most similar and a way to quantify this is through height. The farther the height difference is, the less similar the proteins are. Because the s1.b and the s3.b genes are the same in height and both have a height difference of 350 with the s2.b protein, the two are more similar to each other.

HOMEWORK

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

This is the example output.

```r
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/wp/bd8c6fhs033f5x3l84wb1hx40000gn/T//RtmpdJRtmX/4AKE.pdb exists.
Skipping download
```

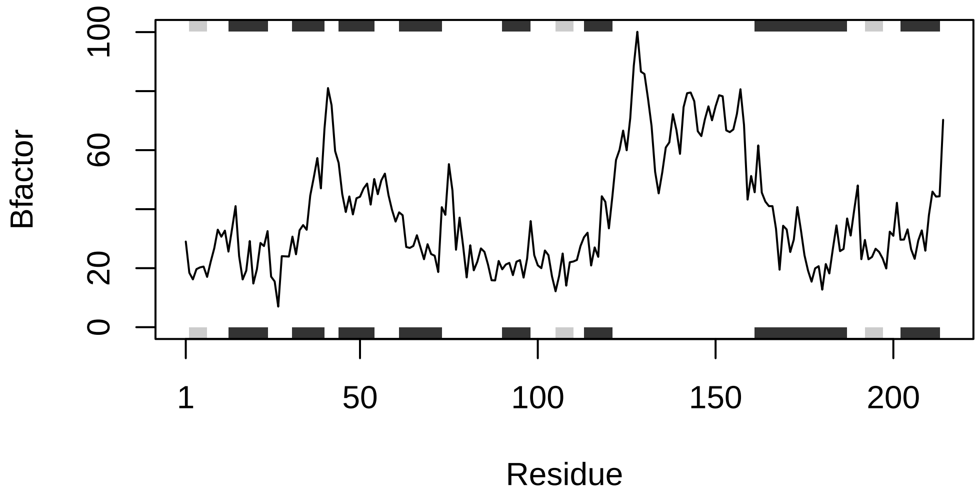
```r
s2 <- read.pdb("1AKE") # kinase no drug
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/wp/bd8c6fhs033f5x3184wb1hx40000gn/T//RtmpdJRtmX/1AKE.pdb exists.
Skipping download

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

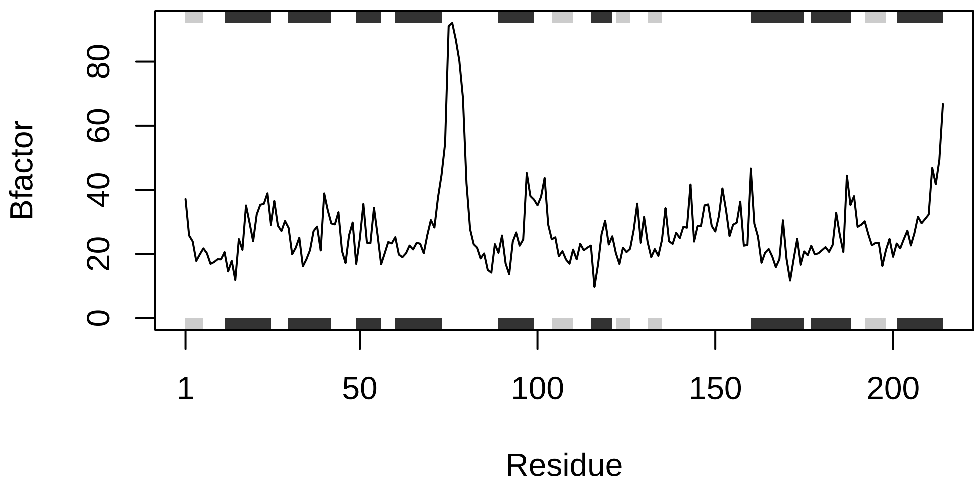
```r
s3 <- read.pdb("1E4Y") # kinase with drug
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/wp/bd8c6fhs033f5x3184wb1hx40000gn/T//RtmpdJRtmX/1E4Y.pdb exists.
Skipping download
```

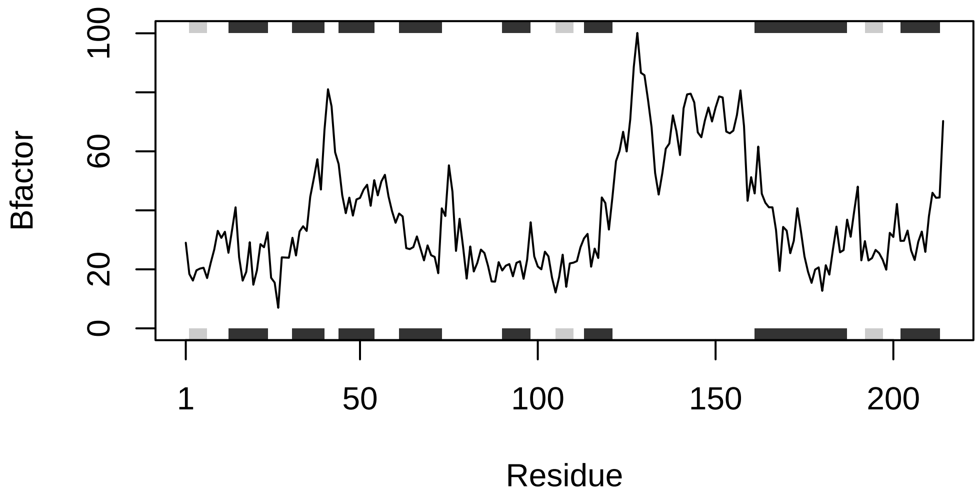
```r
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



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



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



This is the function code.

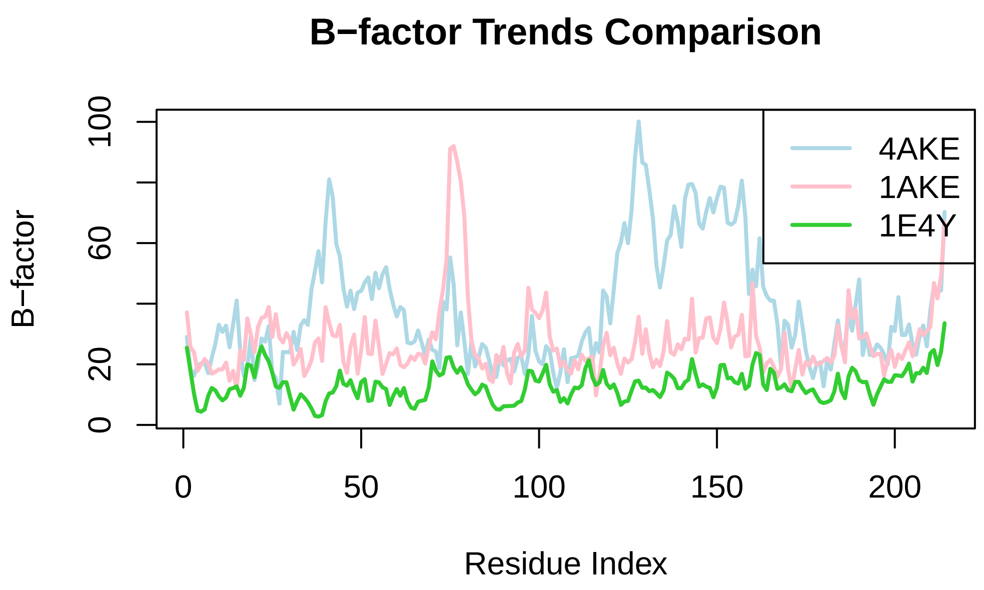
```r
library(bio3d)

# Create a function to read a PDB file and output B-factors for chain A
get_b_factors <- function(pdb_id) {

  # Read the PDB file
  pdb <- read.pdb(pdb_id)

  # Use the `trim()` function to filter to chain A and C-alpha atoms
  chainA <- trim.pdb(pdb, chain="A", elety="CA")

  # Return the B-factors
  return(chainA$atom$b)
}

# List of PDB IDs
pdb_ids <- c("4AKE", "1AKE", "1E4Y")
```

```r
# Initialize an empty list function where you could store B_factors
b_factors_list <- list()

# Loop through each PDB ID to get B-factors
for (id in pdb_ids) {
  b_factors_list[[id]] <- get_b_factors(id)
}
```

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/wp/bd8c6fhs033f5x3184wb1hx40000gn/T//RtmpdJRtmX/4AKE.pdb exists.
Skipping download

  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/wp/bd8c6fhs033f5x3184wb1hx40000gn/T//RtmpdJRtmX/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):
/var/folders/wp/bd8c6fhs033f5x3184wb1hx40000gn/T//RtmpdJRtmX/1E4Y.pdb exists.
Skipping download
```

```r
# Create a combined plot for B-factors for optimal comparison
plot(NULL, xlim=c(1, length(b_factors_list[[1]])),
     ylim=range(unlist(b_factors_list)),
     xlab="Residue Index", ylab="B-factor",
     main="B-factor Trends Comparison")

# Use colors to differentiate the different line plots that represent different B_factors
colors <- c("lightblue", "pink", "limegreen")

# Plot B-factors for each protein
for (i in seq_along(b_factors_list)) {
  lines(b_factors_list[[i]], col=colors[i], type="l", lwd=2)
}
```

```
# Use a legend to simplify read
legend("topright", legend=pdb_ids, col=colors, lty=1, lwd=2)
```


B−factor Trends Comparison
B−factor
Residue Index
4AKE
1AKE
1E4Y