Class 6 Homework

Libby Gilmore

Table of contents

Set-up

```
#install.packages("bio3d")
library(bio3d)
```

Section A. Can you improve this analysis code?

```
# load dataframe
df <- data.frame(a=1:10, b=seq(200,400,length=10),c=11:20,d=NA)
# initial code errors changed
df$a <- (df$a - min(df$a)) / (max(df$a) - min(df$a))
# # this original has an error with df$a being written
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$d)) / (max(df$a) - min(df$d))
```

simplify to core working snippet

```r
# reduced repetitive code with a for loop ------
df_col <- c("a", "b", "c", "d")

# loops through df
for (col in df_col) {
  x <- df[[col]]
  # uses range to normalize each value in each col
  if (is.numeric(x) && !all(is.na(x))) {
    rng <- range(x, na.rm = TRUE)
    if (diff(rng) == 0) {
      df[[col]] <- 0  # avoid divide by zero error
    } else {
      df[[col]] <- (x - rng[1]) / diff(rng)
    }
  }
}

#convert to usable function---------

# normalize_function <- function(df, col){
#   df[[col]] <- (df[[col]] - min(df[[col]])) / (max(df[[col]]) - min(df[[col]]))
# }

# normalizes each column in the data frame
# REQUIRED INPUT: dataframe
# RETURN: updated dataframe with normalized values
normalize <- function(data, columns = names(data)) {
  for (col in columns) {
    if (!col %in% names(data)) next  # skip if column not found
    x <- data[[col]]
    if (is.numeric(x) && !all(is.na(x))) {
      rng <- range(x, na.rm = TRUE)
      if (diff(rng) == 0) {
        data[[col]] <- 0
      } else {
        data[[col]] <- (x - rng[1]) / diff(rng)
      }
    }
  }
```

```
  return(data)
}
```

Test function

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

Section B. Analysis of Protein Drug Interactions

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

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

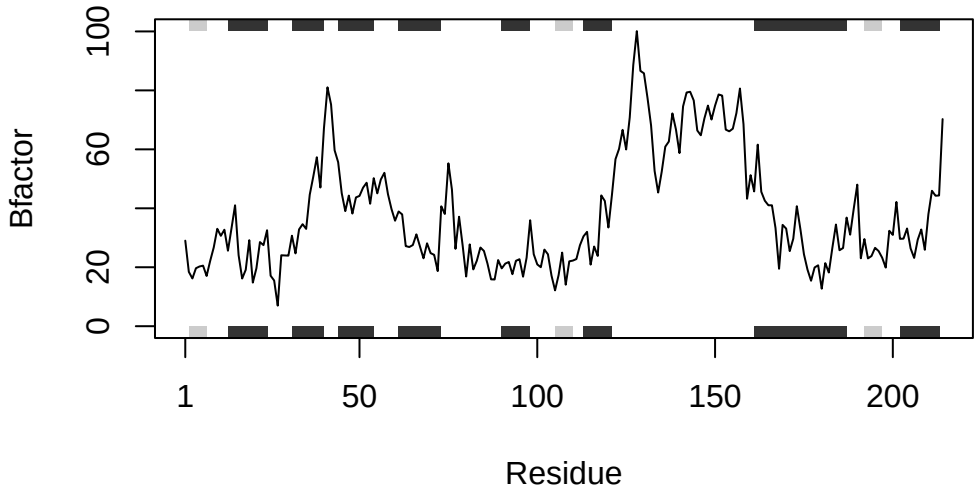
```
s2 <- read.pdb("1AKE")  # kinase no drug
```

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

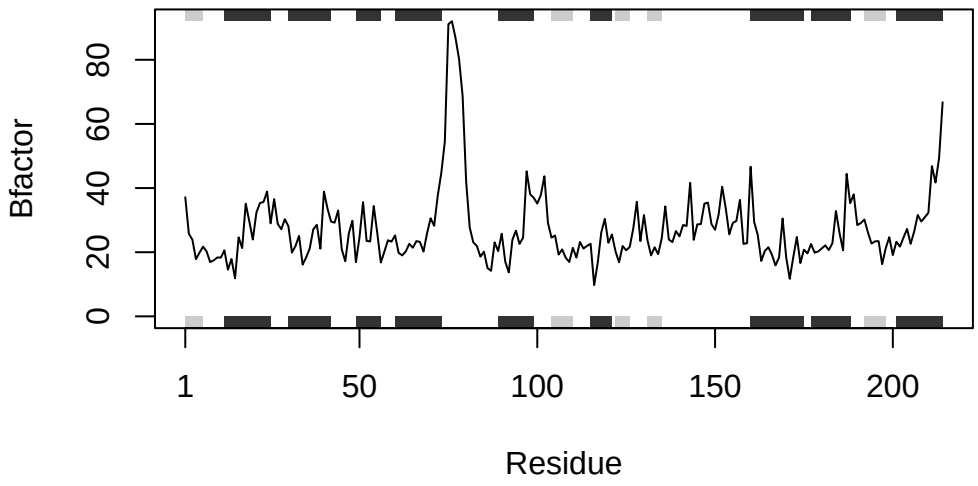
```
s3 <- read.pdb("1E4Y")  # kinase with drug
```

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

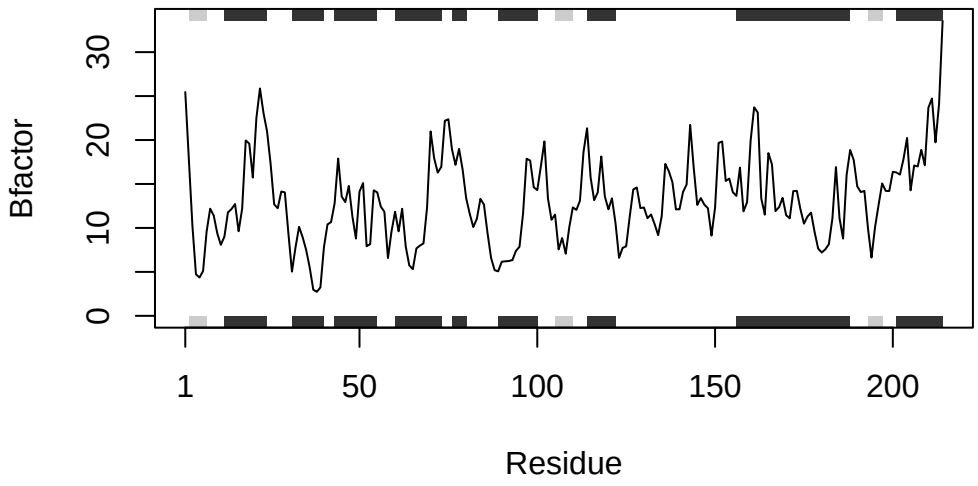
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
# originally wronged, changed from s1 to s3
s3.chainA <- trim.pdb(s3, chain="A", elety="CA")

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")
```



simplified version

```
# Define PDB IDs into vector
pdb_ids <- c("4AKE", "1AKE", "1E4Y")

# Read and trim to atom subset
pdb_list <- lapply(pdb_ids, read.pdb)
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/01/t4pkq4h158zdt81cbt_86rbr0000gn/T//RtmpV0Xxl9/4AKE.pdb exists.
Skipping download
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/01/t4pkq4h158zdt81cbt_86rbr0000gn/T//RtmpV0Xxl9/1AKE.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/01/t4pkq4h158zdt81cbt_86rbr0000gn/T//RtmpV0Xxl9/1E4Y.pdb exists.
Skipping download
```

```
chains <- lapply(pdb_list, trim.pdb, chain = "A", elety = "CA")

# Extract B-factors
b_factors <- lapply(chains, function(x) x$atom$b)
names(b_factors) <- pdb_ids
```

function

```
# INPUT: protein in a string or link
analyze_protein <- function(pdb, chain = "A", elety = "CA", plot = TRUE){
  # Read PDB (can be a file path or a 4-letter PDB ID)
  x <- read.pdb(pdb)

  # Extract only specified chain and C  atoms
  x.chain <- trim.pdb(x, chain=chain, elety=elety)

  # Extract B-factors
  x.bfactor <- x.chain$atom$b

  # Plot B-factors with secondary structure overlay
  plotb3(x.bfactor, sse=x.chain, typ="l", ylab="Bfactor")
}
```

Q1.

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

```
# returns the type of s1, an object generated from read.pdb()
typeof(s1)
```

```
[1] "list"
```

Q2.

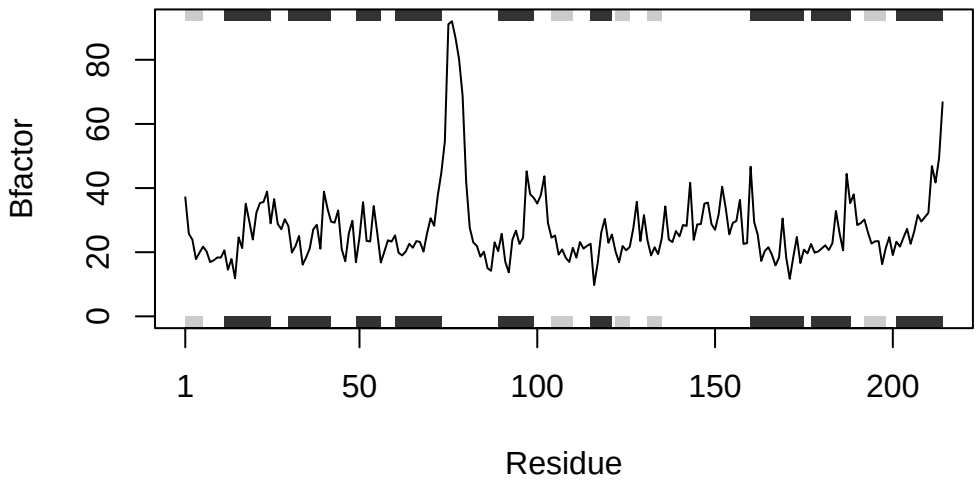
What does the trim.pdb() function do?

```
# trim pdb object to a subset of atoms based on chain id, or atom type.
?trim.pdb
```

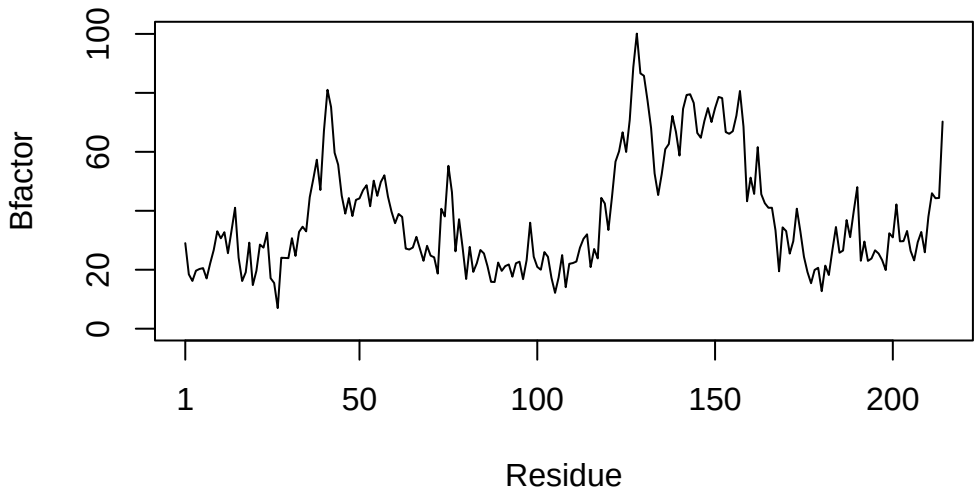
Q3.

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

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



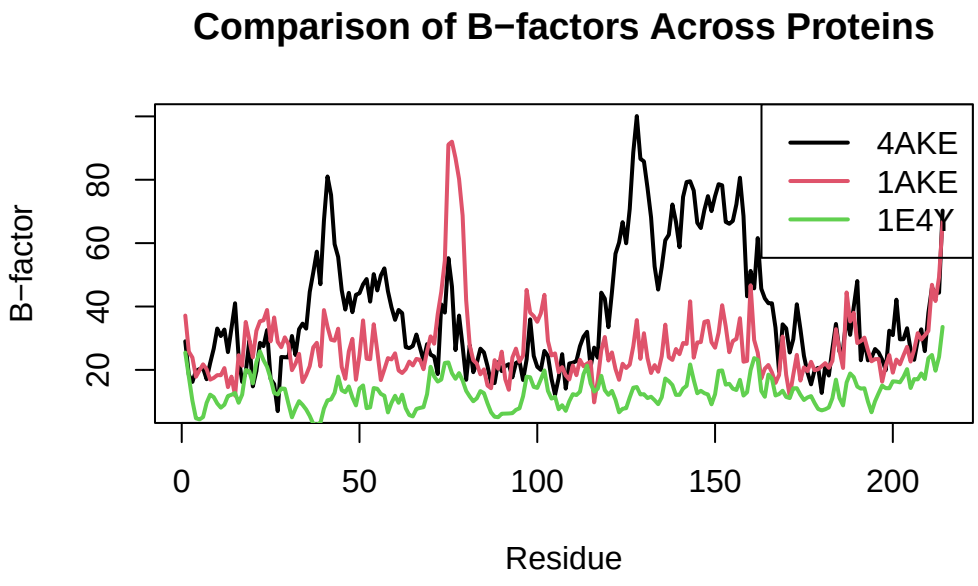

```
# sse argument controls the black and grey squares, so putting NULL or
# FALSE removes them. The squares indicates alpha helices and beta sheets
plotb3(s1.b, sse = NULL, typ = "l", ylab = "Bfactor")
```



Q4.

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

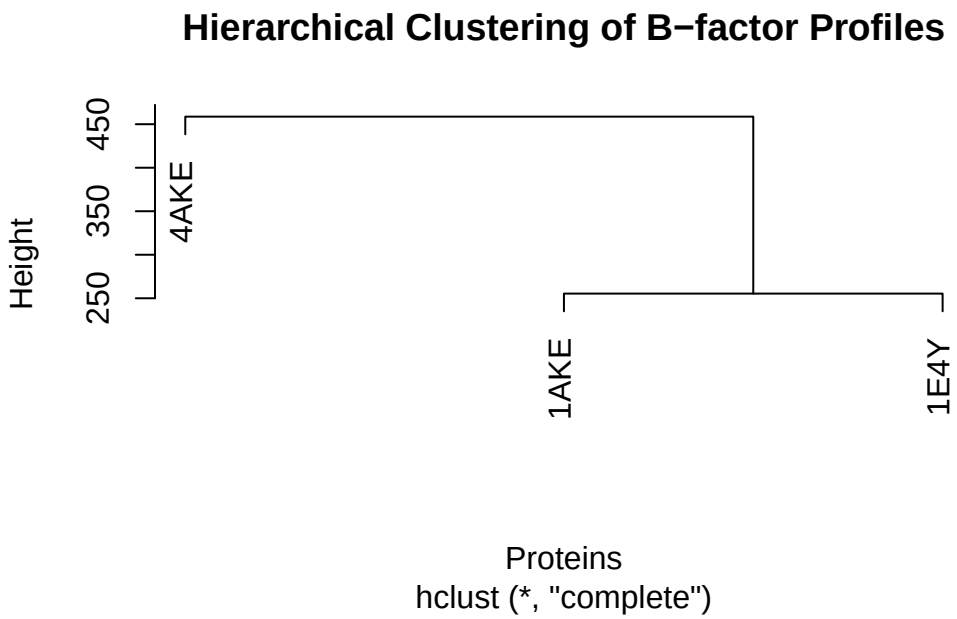
```
# better plot would overlap proteins so you can see differences between their forms

#b_factors is a list of numeric b-factors
# b_factor[[1]] takes the first proteins b-factor value
# plot draws the first line plot
plot(b_factors[[1]], type = "l", col = 1, lwd = 2,
     ylab = "B-factor", xlab = "Residue",
     main = "Comparison of B-factors Across Proteins")

# the for loop draw additional line plots over the inital (done with lines())
# and starts at the second protein
for (i in 2:length(b_factors)) {
  lines(b_factors[[i]], col = i, lwd = 2)
}

# adds legend, and labels which color denotes each line
legend("topright", legend = pdb_ids,
       col = seq_along(b_factors), lty = 1,
       lwd = 2)
```



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

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

# Combine B-factors (must have same length!)
min_len <- min(sapply(b_factors, length))
b_mat <- do.call(rbind, lapply(b_factors, function(x) x[1:min_len]))
rownames(b_mat) <- pdb_ids

# Compute distances and cluster
hc <- hclust(dist(b_mat))
plot(hc, main = "Hierarchical Clustering of B-factor Profiles", xlab = "Proteins")
```



Q6.

How would you generalize the original code above to work with any set of input protein structures? Write your own function starting from the code above that analyzes protein drug interactions by reading in any protein PDB data and outputs a plot for the specified protein.

```r
# function takes in a list of proteins and compares there drug interactions
# required input: pdb_list: vector of strings
# unless specified, will default to chain A, and alpha carbon
analyze_protein_structure <- function(pdb_list, chain = "A",
                                      elety = "CA", plot = TRUE) {
  library(bio3d)

  pdb_list <- lapply(pdb_ids, read.pdb)
  chain_list <- lapply(pdb_list, trim.pdb, chain = chain, elety = elety)
  b_factors <- lapply(chain_list, function(x) x$atom$b)
  names(b_factors) <- pdb_ids

  # Plot overlay
  if (plot) {
    plot(b_factors[[1]], type = "l", col = 1, lwd = 2,
         ylab = "B-factor", xlab = "Residue Index",
         main = paste("B-factor comparison across", length(pdb_ids), "proteins"))
    for (i in 2:length(b_factors)) {
      lines(b_factors[[i]], col = i, lwd = 2)
    }
    legend("topright", legend = pdb_ids,
           col = seq_along(b_factors),
           lty = 1, lwd = 2)
  }

  # Return results for further analysis
  return(list(
    pdb_ids = pdb_ids,
    b_factors = b_factors,
    clustering = hclust(dist(do.call(rbind, lapply(b_factors, function(x) x[1:min(sapply(b_fa
  ))
}

# Example use
results <- analyze_protein_structure(c("4AKE", "1AKE", "1E4Y"))
```

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

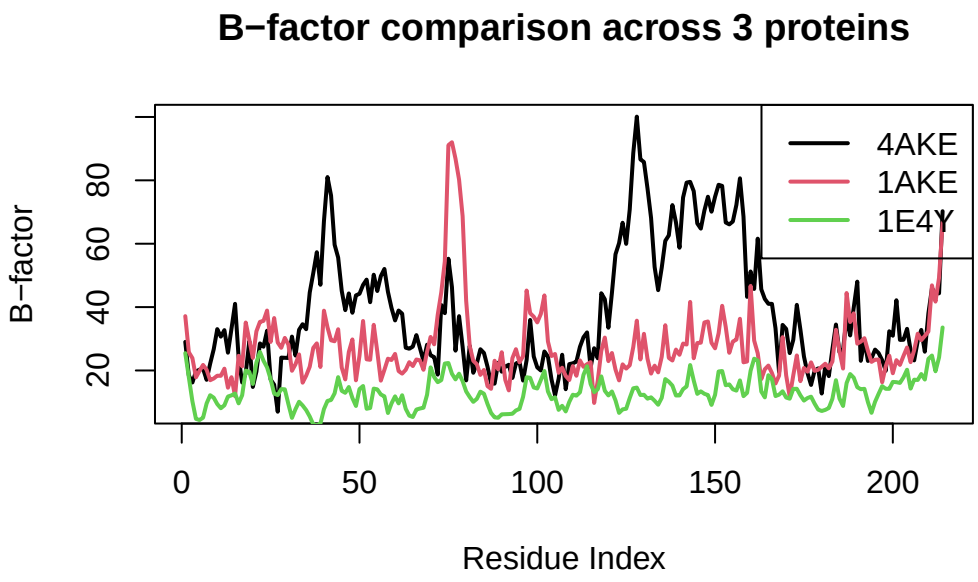
```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/01/t4pkq4h158zdt81cbt_86rbr0000gn/T//RtmpV0Xxl9/4AKE.pdb exists.
Skipping download
```

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

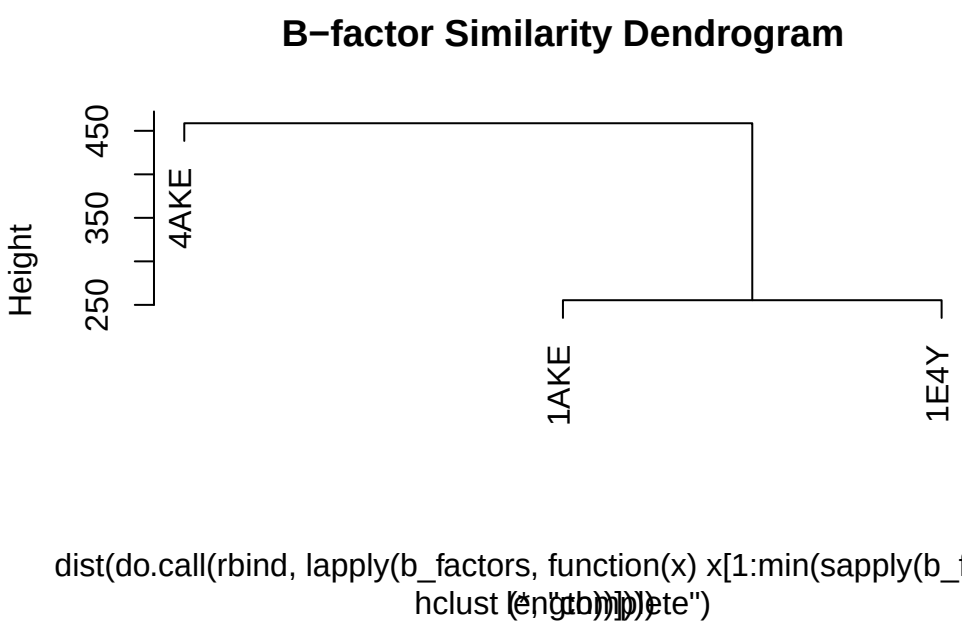
```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/01/t4pkq4h158zdt81cbt_86rbr0000gn/T//RtmpV0Xxl9/1AKE.pdb exists.
Skipping download
```

```
   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/01/t4pkq4h158zdt81cbt_86rbr0000gn/T//RtmpV0Xxl9/1E4Y.pdb exists.
Skipping download
```




```
plot(results$clustering, main = "B-factor Similarity Dendrogram")
```


B−factor Similarity Dendrogram
Height
450
350
250
4AKE
1AKE
1E4Y
dist(do.call(rbind, lapply(b_factors, function(x) x[1:min(sapply(b_factors
hclust (*, "complete")