HWClass6

Jordan Prych (PID: A17080226)

Output of original code:

```
#access bio3d database
library(bio3d)

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

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

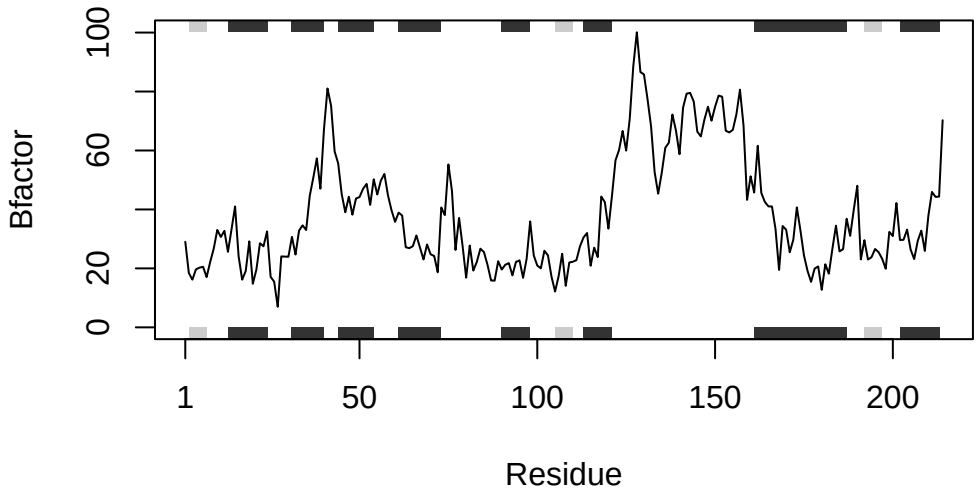
```
 s2 <- read.pdb("1AKE")  # kinase no drug
```

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

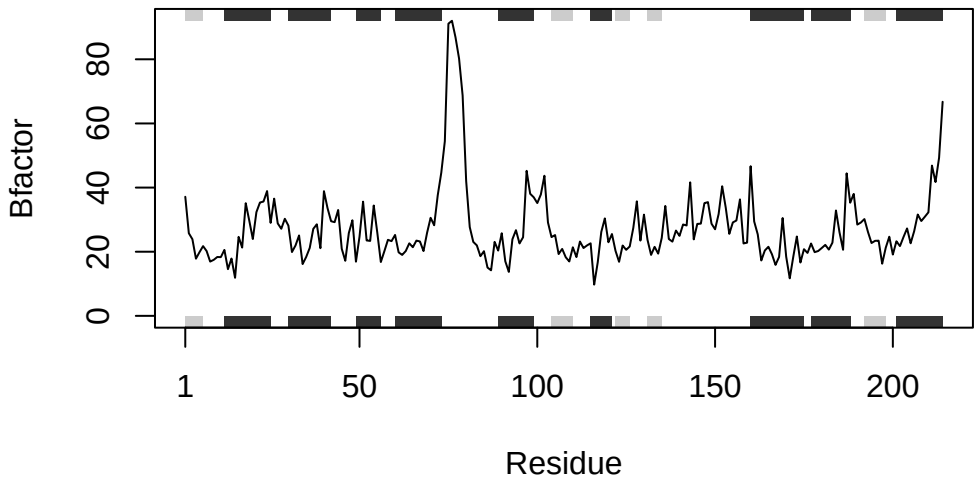
```
 s3 <- read.pdb("1E4Y")  # kinase with drug
```

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

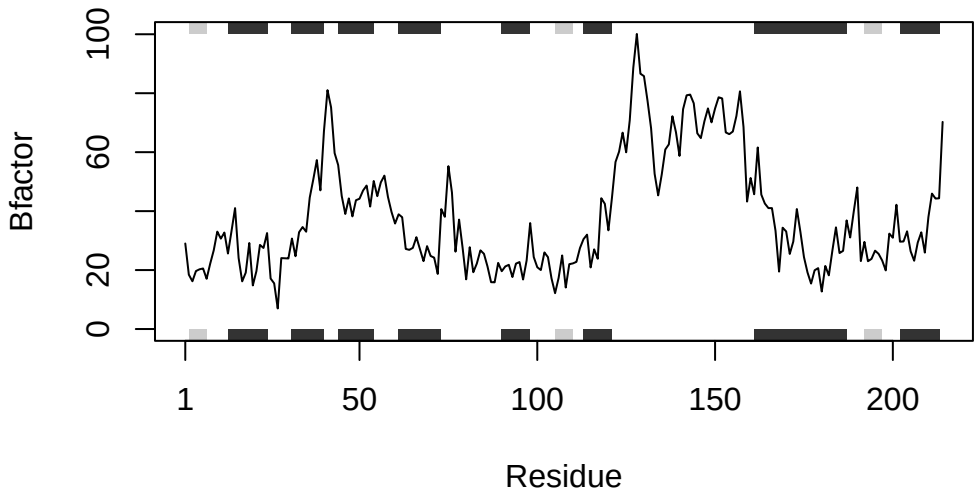
```
#takes the input of previous PDB structure and trims the file to a smaller subset of atoms, l
 s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
 s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
 s3.chainA <- trim.pdb(s1, chain="A", elety="CA")

 #select for atom "b" from "atom" column (selecting beta factor from atom)
 s1.b <- s1.chainA$atom$b
 s2.b <- s2.chainA$atom$b
 s3.b <- s3.chainA$atom$b

#plotting beta factor
 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")
```



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

The `read.pbd()` function calls an online protein database for a specific protein.

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

```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\joelp\AppData\Local\Temp\RtmpILlG66/4AKE.pdb exists. Skipping download


 Call:  read.pdb(file = "4AKE")

   Total Models#: 1
     Total Atoms#: 3459,  XYZs#: 10377  Chains#: 2  (values: A B)

     Protein Atoms#: 3312  (residues/Calpha atoms#: 428)
     Nucleic acid Atoms#: 0  (residues/phosphate atoms#: 0)
```

```
    Non-protein/nucleic Atoms#: 147  (residues: 147)
    Non-protein/nucleic resid values: [ HOH (147) ]

  Protein sequence:
     MRIILLGAPGAGKGTQAQFIMEKYGIPQISTGDMLRAAVKSGSELGKQAKDIMDAGKLVT
     DELVIALVKERIAQEDCRNGFLLDGFPRTIPQADAMKEAGINVDYVLEFDVPDELIVDRI
     VGRRVHAPSGRVYHVKFNPPKVEGKDDVTGEELTTRKDDQEETVRKRLVEYHQMTAPLIG
     YYSKEAEAGNTKYAKVDGTKPVAEVRADLEKILGMRIILLGAPGA...<cut>...KILG

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

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

The `trim.pdb()` function produces a new protein data bank file that provides a subset of information from the original file. Here, only information regarding Chain A and elety variable of the protein is called.

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

```
 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
```

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

-Step 1 - read PDB code using `read.pdb()`. This function will read the file in PBD structures. The output accesses an-online PBD file and reads atom, seqres, helix, sheet, chain, and other variables pertaining to this protein.

-Step 2 - trim PDB structure to a smaller subset of atoms using `trim.pdb()`. This function outputs a trimmed PBD file regarded specified information from the arguments(chain and elety)

-Step 3 - select for atom "b" in chain A from "atom" column. This will select all "b" values in the "atom" column and output these values.

-Step 4 - plot values using `plotb3()` function. This will generate a scatter plot of the beta factor.

```
#access bio3d database
library(bio3d)
#input to function is protein PDB file to read
plot.pdb <- function(pdb.protein) {
#read in PDB structures
   s <- read.pdb(pdb.protein)
#takes the input of previous PDB structure and trims the file to a smaller subset of atoms, l
  s.chainA <- trim.pdb(s, chain="A", elety="CA")
#select for atom "b" from "atom" column (selecting beta factor from atom)
  s.b <- s.chainA$atom$b
#plotting beta factor
  plotb3(s.b, sse=s.chainA, typ="l", ylab="Bfactor")
}
```

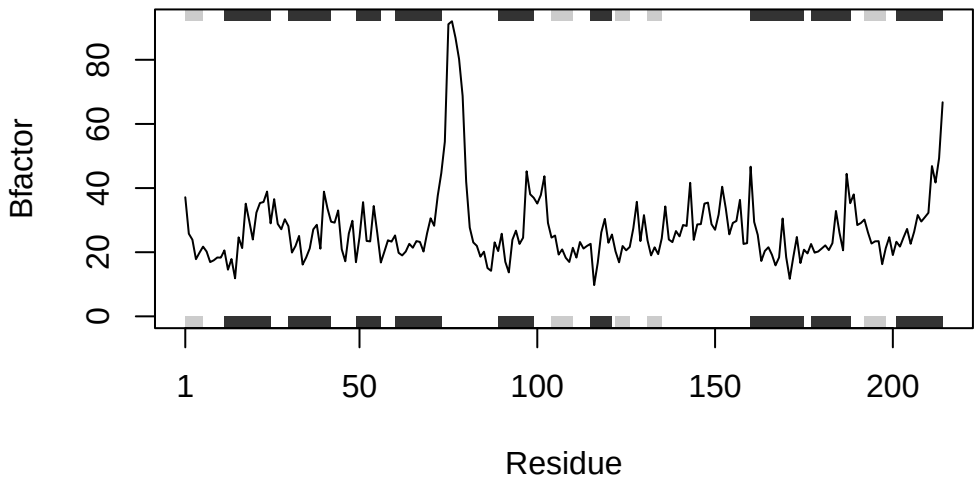
Testing generated function:

```
plot.pdb("1AKE")
```

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

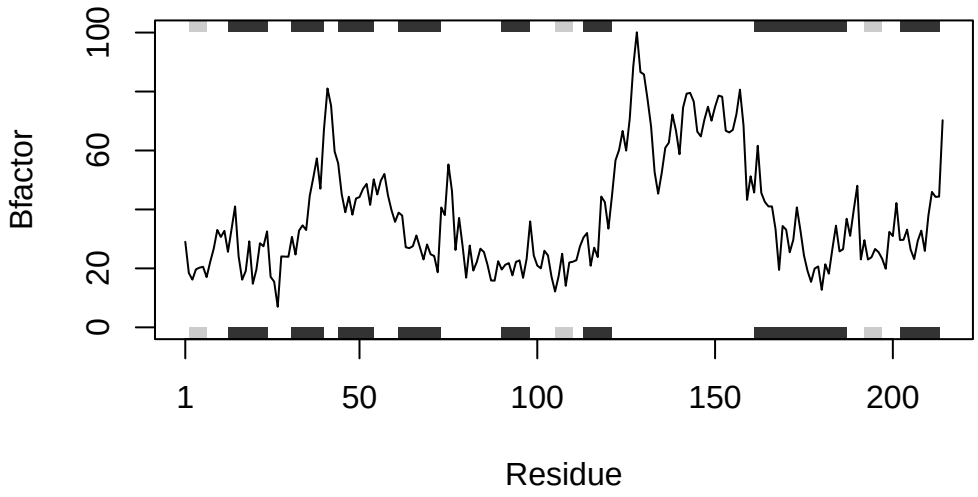
```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\joelp\AppData\Local\Temp\RtmpILlG66/1AKE.pdb exists. Skipping download
```

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




```
plot.pdb("4AKE")
```

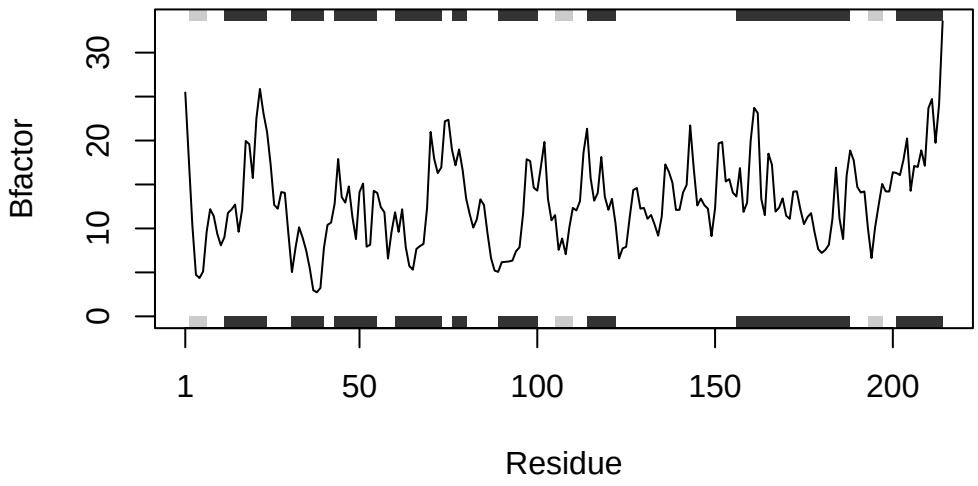
```
  Note: Accessing on-line PDB file

Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\joelp\AppData\Local\Temp\RtmpILlG66/4AKE.pdb exists. Skipping download
```




```
plot.pdb("1E4Y")
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\joelp\AppData\Local\Temp\RtmpILlG66/1E4Y.pdb exists. Skipping download
```


Bfactor
0
10
20
30
1
50
100
150
200
Residue