Generalizing Functions in R

Andy Hsu

More Function Practice

Taking this code, a function can be written to apply to any protein structure.

```
library(bio3d)
s1 <- read.pdb("4AKE")
```

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

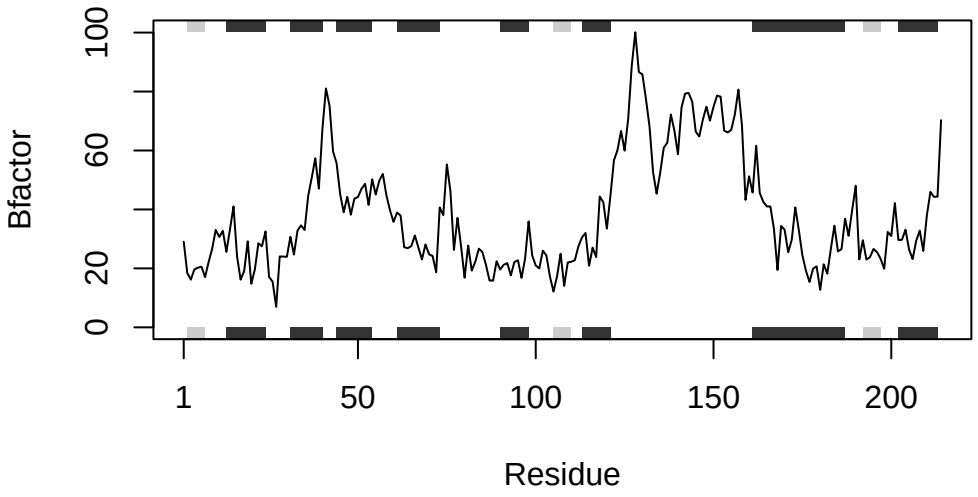
```
s2 <- read.pdb("1AKE")
```

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

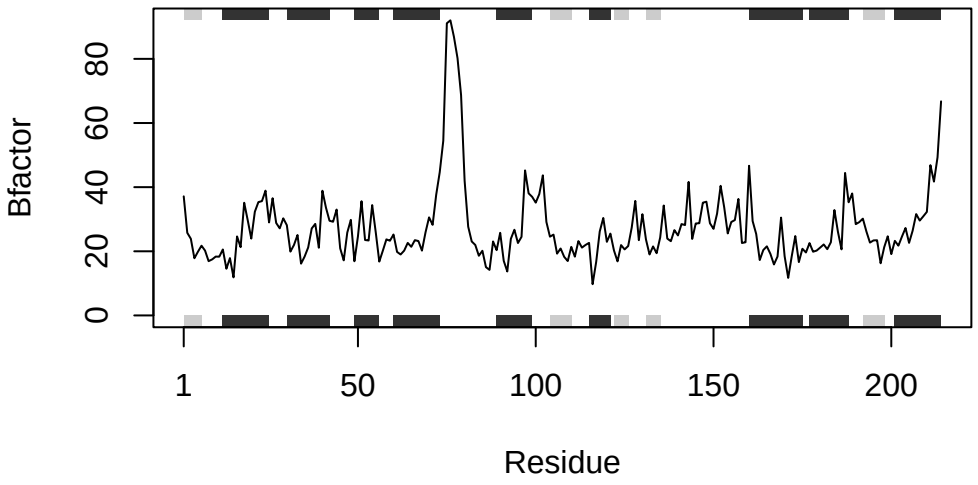
```
s3 <- read.pdb("1E4Y")
```

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

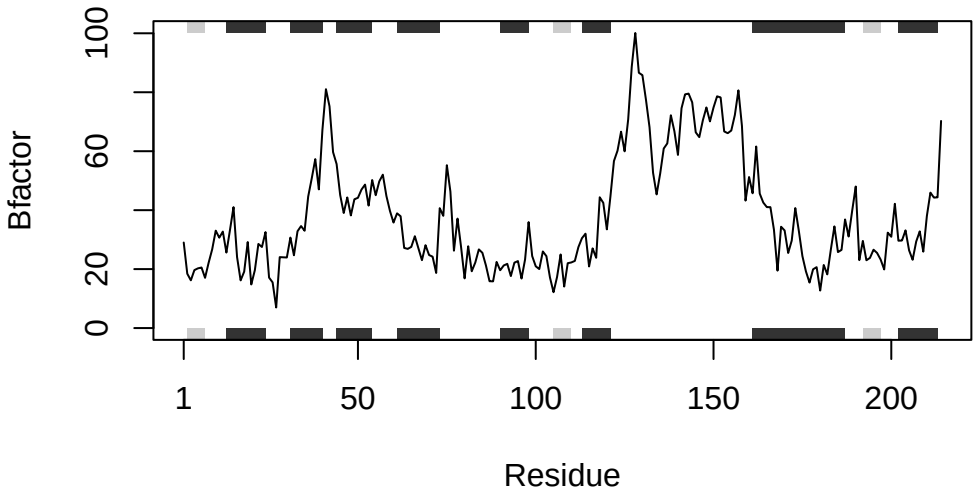
```
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")

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")
```



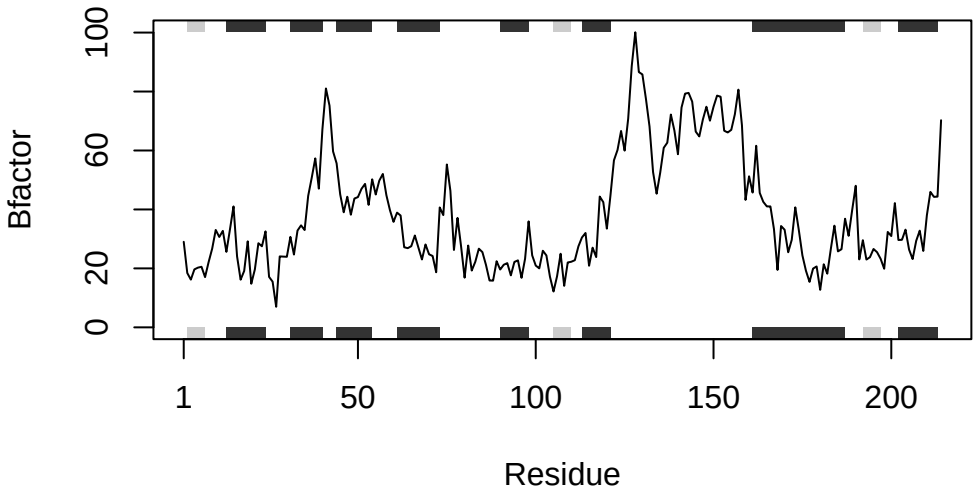
First, we condense this code down to the snippet for 1 input.

```
library(bio3d)
s1 <- read.pdb("4AKE")
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\andyp\AppData\Local\Temp\RtmpuOcu3Y/4AKE.pdb exists. Skipping download
```

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



Then, we map this to a function, using more general variables.

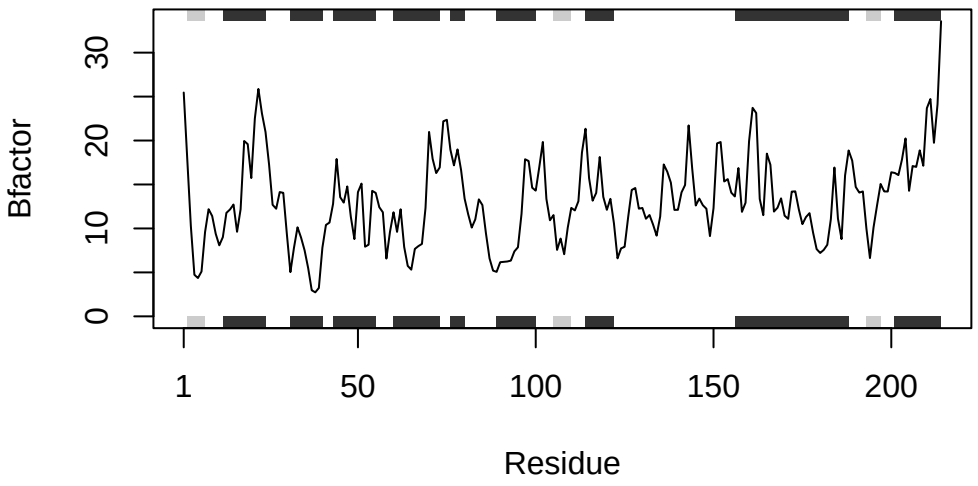
```
BfacPlot <- function(x) {
  y <- read.pdb(x)
  z <- trim.pdb(y,chain="A",elety="CA")
  w <- z$atom$b
  plotb3(w,sse=z,typ="l",ylab="Bfactor")
}
```

To use this function, simply enter `BfacPlot(x)`, with x substituted to any 4-character PDB ID. The function will return a line plot of Bfactor per residue.

```
BfacPlot("1E4Y")
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
C:\Users\andyp\AppData\Local\Temp\RtmpuOcu3Y/1E4Y.pdb exists. Skipping download
```


Bfactor
0
10
20
30
1
50
100
150
200
Residue