Class 06 - Functions in R

Ellice Wang (PID: A16882742)

Today we will get more exposure to functions in R. We will call functions to do all our work and today we will learn how to write our own.

A first silly function

Note that arguments 2 and 3 have default values because we set y=0 and z=0, so we don't have to supply them when we call the function.

```
add <- function(x, y=0, z=0) {
  x+y
}
```

```
add(1,1)
```

```
[1] 2
```

```
add(1,c(10,100))
```

```
[1]  11 101
```

```
add(100)
```

```
[1] 100
```

```
add(100,10,5)
```

```
[1] 110
```

A second more fun function

Let’s write a function that genereates random nucleotide sequences.

We can make use of the in-built ‘sample()’ function in R to help us here.

```r
sample(x=1:10, size=11, replace=T)
```

```
 [1]  1 10  3  4  8  6  7  1  2  2  3
```

> Q. Can you use ‘sample()’ to generate a random nucleotide sequence of length 5.

```r
sample(x=c('A', 'T', 'C', 'G'), size=5, replace=T)
```

```
[1] "G" "G" "T" "A" "C"
```

> Q. Generate a function ‘generate_dna()’ that makes a nucleotide sequence of a user defined length

Every function in R has at least 3 things:

- a **name** (in our case ‘generate_dna’)
- one or more **input arguments** (length of sequence we want)
- a **body** (R code that does the work)

```r
generate_dna <- function(dna_len=5) {
  sample(x=c("A", "T", "C", "G"), size=dna_len, replace=TRUE)
}

# test to see if the function works
generate_dna(50)
```

```
 [1] "A" "A" "A" "G" "T" "A" "A" "T" "C" "T" "C" "T" "A" "G" "G" "T" "A" "C" "G"
[20] "C" "G" "G" "T" "G" "A" "A" "T" "C" "C" "G" "G" "C" "A" "A" "G" "A" "G" "C"
[39] "G" "T" "T" "A" "A" "C" "A" "C" "T" "T" "T" "C"
```

> Q. Can you write a ‘generate_protein()’ function that returns amino acid sequence of a user requeseted length?
>
> ‘install.packages(“bio3d”)’

```r
# aa_codes <- c("C", "Q", "N", "T", "S", "G", "A", "P", "M", "I", "L", "V",
#               "F", "Y", "W", "D", "E", "H", "K", "R")

# instead of typing out all of the amino acids, can you the bio3d shortcut
generate_protein <- function(protein_len=5){
  aa_codes <- bio3d::aa.table$aa1[1:20]
  paste(sample(aa_codes, protein_len, replace = TRUE), collapse="")
}

# check to see if it works; random length of 30
generate_protein(30)
```

```
[1] "KLRVADCERTNDDIRNCIQVFRTVIWQAHL"
```

want output to be a concatenated string not a vector with one amino acid per element

```r
bases <- c("A", "G", "C", "T")
paste(bases, collapse="")
```

```
[1] "AGCT"
```

> Q. Generate protein sequences from length 6 to 12?

We can use the useful utility function ‘sapply()’ to help us “apply” our function over all the values 6 to 12

```r
ans <- sapply(6:12, generate_protein)

cat(paste(">ID", 6:12, sep="", "\n", ans, "\n"))
```

```
>ID6
GSIMTF
 >ID7
HRSTDVT
 >ID8
YVILFSAV
 >ID9
HIQPQKYLI
 >ID10
VYEHHSAIIG
```

```
 >ID11
VMFRCYFAWHE
 >ID12
YMLEIKEELNSR
```

> Q. Are any of these sequences unique in nature? - i.e. never found in nature. We can search "refseq-protein" and look for 100% Identity and 100% coverage matcheswith BLASTPp

Some of these sequences are found in nature. "AFELCW" mapped to different proteins like: dynein heavy chain [Kickxella alabastrina], leucine-rich repeat protein [Paraprevotella xylaniphila], among others. "FRRIAFM" mapped to multiple proteins like: receptor-type adenylate cyclase a [Leishmania major strain Friedlin],

```r
library(bio3d)
```

```
Warning: package 'bio3d' was built under R version 4.4.1
```

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

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

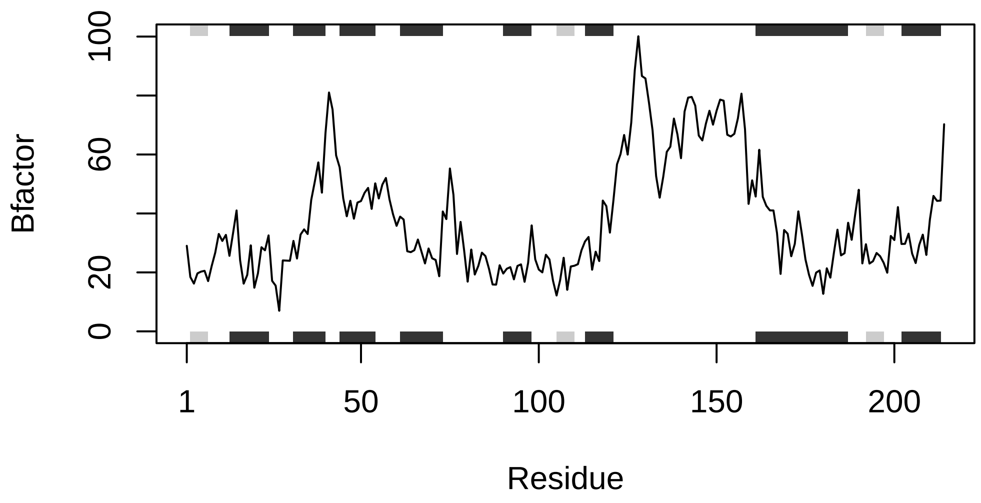
```r
s2 <- read.pdb("1AKE") # kinase no drug
```

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

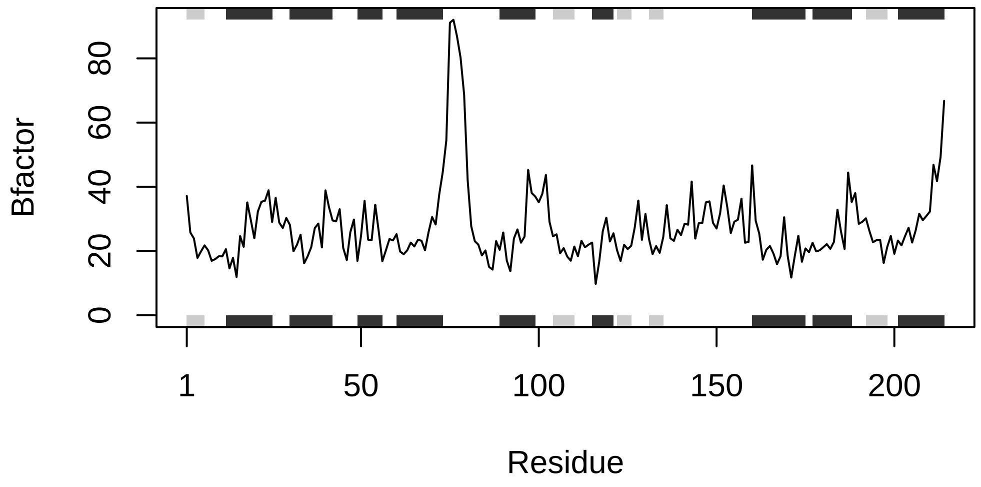
```r
s3 <- read.pdb("1E4Y") # kinase with drug
```

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

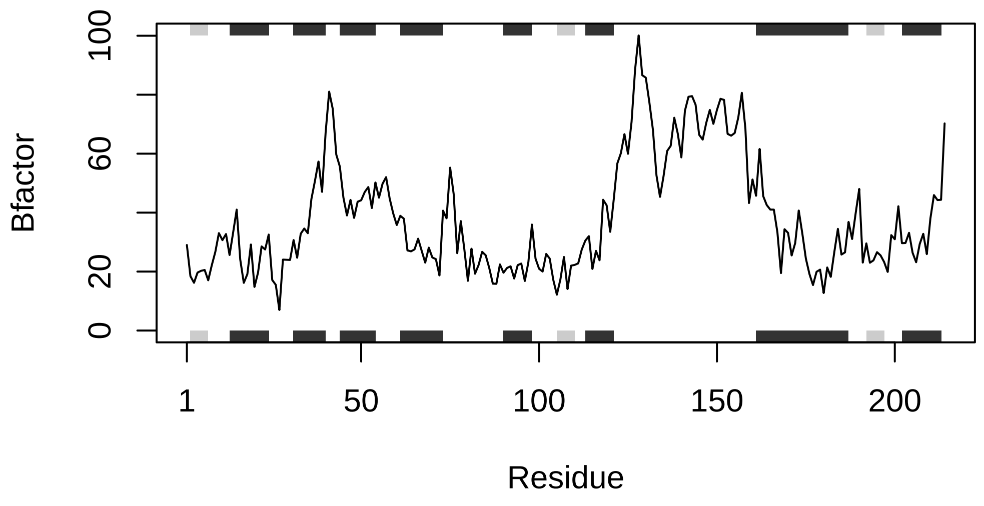
```r
s1.chainA <- trim.pdb(s1, chain="A", elety="CA")
s2.chainA <- trim.pdb(s2, chain="A", elety="CA")
s3.chainA <- trim.pdb(s1, chain="A", elety="CA")
s1.b <- s1.chainA$atom$b
s2.b <- s2.chainA$atom$b
s3.b <- s3.chainA$atom$b
plotb3(s1.b, sse=s1.chainA, typ="l", ylab="Bfactor")
```



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



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

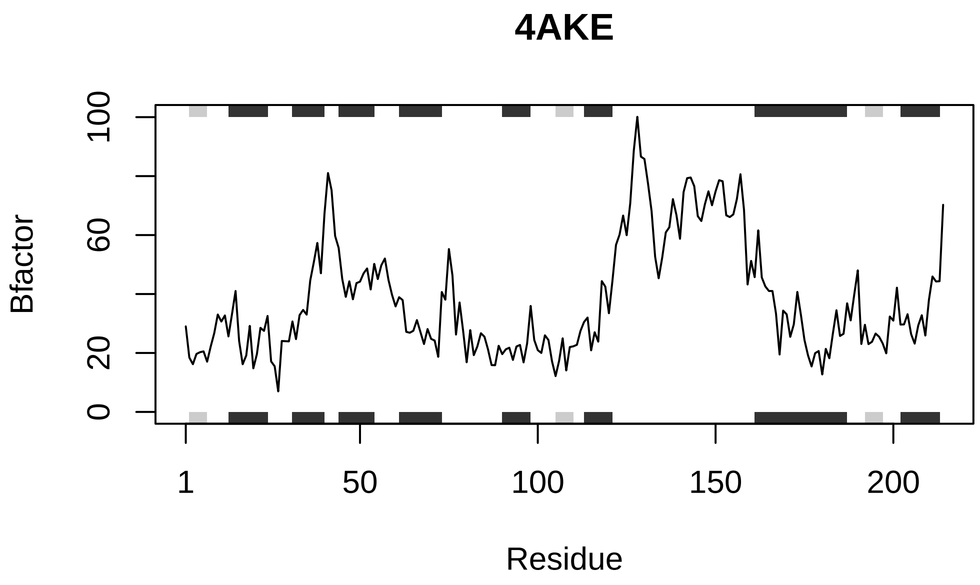


Q6. How would you generalize the original code above to work with any set of input protein structures?

```r
# load in bio3d package
library(bio3d)

compare_bfactor <- function(kinase_input) {
  # input is a 4 character protein code
  # output is a line graph depicting the B factor trends of the kinase

  # load in protein database dataframe of inputed kinase
  pdb_df <- read.pdb(kinase_input)

  # select the A chain
  pdb_df.chainA <- trim.pdb(pdb_df, chain="A", elety="CA")

  # select the b factor of the A chain
  pdb_df.b <- pdb_df.chainA$atom$b

  # plot the Bfactor of the kinase
  plotb3(pdb_df.b, sse=pdb_df.chainA, typ="l", ylab="Bfactor", main=kinase_input)
}

# confirm that it works
compare_bfactor("4AKE")
```

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

```
Warning in get.pdb(file, path = tempdir(), verbose = FALSE):
/var/folders/hz/3cdfl2jj3qv7fyz_dbg0rc2r0000gn/T//RtmpRUB5ZE/4AKE.pdb exists.
Skipping download
```


4AKE
Bfactor
Residue