Class 6: R functions

Jiayi Zhou (PID: A17856751)

All functions in R have at least 3 things:

- A name, we pick this and use it to call the function.
- Input arguments, there can e multiple comma separated inputs to the function.
- The body, lines of R code that do not work of the function.

Our first wee function:

```r
add <- function(x,y=1) {
x + y
}
```

Let' test our function

```r
add(c(1,2,3), y=10)
```

```
[1] 11 12 13
```

```r
add(10,100)
```

```
[1] 110
```

A sencond function

Let's try something more interesting. Make a sequence generation tool. The sample() function could be useful here.

```r
sample(1:10, size = 3)
```

```
[1] 1 5 4
```

change this to work with the nucleotides A C G and T and return 3 of them

```r
n <- c("A","C","G","T")
sample(n,size=15, replace = TRUE)
```

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

Turn this snipet into a function that returns a user specified length dna sequence. Let's call it *generate_dna()*…***

```r
generate_dna <- function(len=10, fasta=TRUE) {
n <- c("A","C","G","T")
v <- sample(n,size=len, replace = TRUE)
#Make a single element vector
s <- paste(v, collapse="")
cat("Well done you!\n")
if (fasta) {
return(s)
} else {
return(v)
}
}
generate_dna(fasta=TRUE)
```

```
Well done you!
```

```
[1] "CCTAGTCTTC"
```

```r
generate_dna(fasta=FALSE)
```

```
Well done you!
```

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

I want the option to return a single element character vector with my sequence all together like this: “GGAGTAC”

```r
s <- c("A","C","G","T")
s
```

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

```r
paste(s, collapse = "")
```

```
[1] "ACGT"
```

```r
generate_dna <- function(len = 10, fasta = FALSE) {
n <- c("A", "C", "G", "T")
v <- sample(n, size = len, replace = TRUE)
s <- paste(v, collapse = "")
cat("Well done you!\n")
if (fasta) {
return(s)
} else {
return(v)
}
}
```

```r
lookatme <- function (size=15, fasta=FALSE){
n=c("A","C","G","T")
seq <- sample(n,size=size, replace=TRUE)
if (fatsa) {
return(paste(seq,colapse=""))
}else{
return(seq)
}
}
```

A more advanced example

Make a third function that generates protein sequence of a user specified length and format.

```r
generate_protein <- function(size = 15, fasta = TRUE) {
aa <- c("A","R","N","D","C","Q","E","G","H","I",
"L","K","M","F","P","S","T","W","Y","V")
seq <- sample(aa, size = size, replace = TRUE)
if (fasta) {
return(paste(seq, collapse = ""))
} else {
return(seq)
}
}
```

Try this out…

```r
generate_protein(10)
```

```
[1] "MVQELFTVCL"
```

> Q. Generate random protein sequences between lengths 5 and 12 amino acids

```r
generate_protein(5)
```

```
[1] "WCWTE"
```

```r
generate_protein(6)
```

```
[1] "TKEVWC"
```

One approach is to do this by brute force calling our function for each length 5 to 12. Another approach is to write a `for()` loop to itterate over he input valued 5 to 12 A very useful third R specific approach is to use the `sapply()` function

```r
seq_lengths <- 6:12
for (i in seq_lengths){
cat(">",i,"\n")
cat(generate_protein(i))
cat("\n")
}
```

```
> 6
WDGHTR
> 7
NMPEKKR
> 8
EIMMMKPS
> 9
NVQKPREFL
> 10
WNPAHQTLSP
> 11
EFKWEDPPDPM
> 12
AADWPTFPWIHN
```

```r
sapply(5:12, generate_protein)
```

```
[1] "FSLMW"        "PMVQMK"       "SELNYSY"      "IWSGRTLS"     "LKDKYHHEY"
[6] "CELCWYDAPE"   "MTKAIEHEKPM"  "HTPNTCPWKNAC"
```

> **Key-point**:Writing functions in R is doable but not easiest thing. Starting with a working snippet of code then using LLM tools to improve and generalize your function is a productive approach.