Class 6: R functions

Blinda Sui (PID: A17117043)

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 be multiple comma seperated inputs to the function. The **body**, lines of R code that do the work of the function.

Our first wee function:

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

Let's test our function

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

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

```
add(10)
```

```
[1] 11
```

```
add(10, c(10, 120))
```

```
[1]  20 130
```

A second function

Let's try something more interesting. Make a sequence generation tool.

The 'sample()' function could be useful here.

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

```
[1] 8 7 4
```

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

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

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

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

```
generate_dna <- function(len=10, fasta=FALSE){
  n <- c("A", "G", "C", "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(5)
```

```
Well done you!
```

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

```
s <- generate_dna(15)
```

```
Well done you!
```

```
s
```

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

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

```
s
```

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

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

```
[1] "CGGCGTGGTCTTTCA"
```

```
generate_dna(10, fasta=TRUE)
```

```
Well done you!
```

```
[1] "TACAGTGTCC"
```

```
generate_dna(10, fasta=FALSE)
```

```
Well done you!
```

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

A more advanced example

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

```
generate_protein <- function(size = 15, fasta = TRUE) {
  aa <- c("A","R","N","D","C","E","Q","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...

```
generate_protein(10)
```

```
[1] "RMQNNFTEQD"
```

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

```
generate_protein(5)
```

```
[1] "SFPVW"
```

```
generate_protein(6)
```

```
[1] "AQSPFL"
```

One approach is to do this by brute force calling our unction for each lenght 5 to 12.

Another approach is to write a 'for()' loop to itterate over the input valued 5 to 12.

A very useful third R specific approach is to use the 'sapply()' function.

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

```
> 6
EWSAGQ
> 7
IPACPHH
> 8
CPVSMSKY
> 9
NFQLLGLMQ
> 10
DAMQHIVCPN
> 11
QWVFDNVEKHK
> 12
HVWFAAAYDFKP
```

```
sapply(6:12, generate_protein)
```

```
[1] "HHQGNR"       "WRCHYAH"      "FEYKKILG"     "ETMSPYVDV"    "VQFFNMQKHL"
[6] "TVVWSNFKGTD"  "YWFLYHVFLDTI"
```

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