Class 6: R function

Charlize Molitor (PID: A18515740)

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(10,100)
```

```
[1] 110
```

A second function

Let's try something more intresting. Make a sequence generational tool.

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

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

```
[1] 5 8 6
```

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

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

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

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", "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(5)
```

```
Well done you!
```

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

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

```
Well done you!
```

```r
s
```

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

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

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

```
Well done you!

[1] "GGAGCGTGTT"
```

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

```
Well done you!

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

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"
  seq <- sample(aa, size = size, replace = TRUE)

  if (fasta) {
    return(paste(seq,collapse = ""))
  } else {
    return(seq)
  }
}
```

Try this out…

```r
generate_protein(10)
```

```
[1] "MNSARDIYDQ"
```

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

```r
generate_protein(5)
```

```
[1] "CHDDF"
```

```r
generate_protein(6)
```

```
[1] "EGHQAY"
```

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 the input valued 5 to 12

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

```r
seq_lengths <- 5:12
for (i in seq_lengths) {
  cat(">", i, "\n")
  cat(generate_protein(i))
  cat("\n")
}
```

```
> 5
YITRD
> 6
IKKIFV
> 7
MPMRHNG
> 8
VLISGLVL
> 9
WHCPRCPQM
> 10
HCARVRARPF
```

```
> 11
TKMYNMHTFNW
> 12
LTLIWLQTSIMW
```

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

```
[1] "YLNIL"         "WSGDDY"        "WISTQGE"       "HPMHIFKP"      "EAYNIHGLV"    
[6] "QAHKYRAWAN"    "KYCDHLFQCIT"   "LYCCNQTCVVQH"
```

> **Key-Point**: Writing function 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 in a productive approach.