Class 6: R Functions

Le, Sarah (PID: A18518276)

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 interesting. Make a sequence generation tool.

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

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

```
[1] 6 8 4
```

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] "T" "T" "C" "A" "A" "T" "A" "C" "G" "G" "T" "C" "A" "T" "C"
```

Turn this snipet into a function that returns a user specific 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] "C" "T" "C" "A" "T"
```

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

```
Well done you!
```

```
s
```

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

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

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

```
Well done you!
```

```
[1] "TGGTTGGCTA"
```

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

```
Well done you!
```

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

A more advanced example.

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

```
generate_protein <- function(len=10, 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 = len, replace = TRUE)

  if (fasta) {
    return(paste(seq, collapse = ""))
  } else {
  return(seq)
  }
}
generate_protein(10)
```

```
[1] "WKFTKSCCML"
```

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

```
generate_protein(5)
```

```
[1] "RNGFN"
```

```
generate_protein(6)
```

```
[1] "DKHIHA"
```

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 iterate over the input valued 5 to 12.

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

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

```
>5
GSLCH
>6
IMNIPT
>7
GFQAGGM
>8
LQIQSLAQ
>9
QGSSCWACG
>10
FVKMEWKALN
>11
CHTITNQRENN
>12
KTAHRAAKGMPD
```

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

```
[1] "KSCDW"        "ATGSNL"       "TLLCMWS"      "KSTVMGTI"     "NAYEYICRG"
[6] "CVNDMFHNST"   "MPTQNSPCSCH"  "YLQIGYKPGVSM"
```

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.