R Functions Lab Class 6

Carolina Merino

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) {
  x + 1
}
```

Let's test our function

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

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

```
[1] 16
```

```
add(10)
```

```
[1] 10
```

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 6 8
```

Change this to work with nucleotides A C G and T and return a 3 of them

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

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

Turn this snippet into a function that returns a user specificied length dna sequence. Let's call it `generate_dna()`…

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

```r
generate_dna <- function(length) {
  # Define nucleotides
  nucleotides <- c("A", "C", "G", "T")

  # Sample nucleotides with replacement
  dna_vector <- sample(nucleotides, size = length, replace = TRUE)

  # Optional: print a message
  cat("Well done you!\n")

  # Combine into a single string
  dna_sequence <- paste(dna_vector, collapse = "")

  # Return the sequence
  return(dna_sequence)
}

# Example usage:
generate_dna(15)
```

```
Well done you!

[1] "GATTGTGATAGACAC"
```

```
# See data for 15 dna sequence and for 25 dna sequence
generate_dna(15)
```

```
Well done you!

[1] "GTGCGGCCTCCCCCC"
```

```
generate_dna(25)
```

```
Well done you!

[1] "CCAGCTACGACAGGCCTGGAATGGG"
```

Now to generate a DNA sequence as either a single string or a vector of nucleotides

```
# Function to generate a random DNA sequence
# fasta = TRUE returns a single string, FALSE returns a vector
generate_dna <- function(length, fasta = TRUE) {
  nucleotides <- c("A", "C", "G", "T")
  dna_vector <- sample(nucleotides, size = length, replace = TRUE)

  if(fasta) {
    paste(dna_vector, collapse = "")  # single string
  } else {
    dna_vector  # vector of letters
  }
}

# Examples:
generate_dna(10)                  # default fasta = TRUE, returns "GGATACGCTA"
```

```
[1] "ACGCTGCGCG"
```

```
generate_dna(10, fasta = FALSE)   # returns vector c("G","G","A",...)
```

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

A more advanced example:

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

```
# Function to generate a random protein sequence
# fasta = TRUE returns a single string, FALSE returns a vector of amino acids
generate_protein <- function(length, fasta = TRUE) {
  # Define the 20 standard amino acids
  amino_acids <- c("A", "R", "N", "D", "C", "Q", "E", "G",
                   "H", "I", "L", "K", "M", "F", "P", "S",
                   "T", "W", "Y", "V")

  # Sample amino acids with replacement
  protein_vector <- sample(amino_acids, size = length, replace = TRUE)

  # Return as single string or vector
  if(fasta) {
    paste(protein_vector, collapse = "")  # combine into one string
  } else {
    protein_vector  # return as vector of letters
  }
}

# Examples:
generate_protein(10)              # returns single string like "MKTAGLIPRY"
```

```
[1] "FHNYVKEHNA"
```

```
generate_protein(10, fasta = FALSE) # returns vector c("M","K","T",...)
```

```
 [1] "V" "D" "E" "V" "K" "A" "D" "K" "V" "I"
```

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

Questions to answer:

```
generate_protein(5)
```

```
[1] "FSCHM"
```

```
generate_protein(6)
```

```
[1] "SSLLVP"
```

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

This format, takes a lot of effort..

```
# Sequence of lengths
seq_lengths <- 6:12

# Loop over each length
for (i in seq_lengths) {
  dna_seq <- generate_dna(i)   # call your generate_dna() function
  cat("Length", i, "->", dna_seq, "\n")
}
```

```
Length 6 -> GGAATC
Length 7 -> CACAATT
Length 8 -> GTAACACA
Length 9 -> TAGTAGCGT
Length 10 -> GCGAGGTGAC
Length 11 -> CATTACGCTGT
Length 12 -> CCTCGTTGCCGC
```

Let's do the `sapply()`

```
# Sequence of lengths
seq_lengths <- 6:12

# Generate DNA sequences as single strings
dna_sequences <- sapply(seq_lengths, function(len) generate_dna(len, fasta = TRUE))

# Show results
dna_sequences
```

```
[1] "CGCCCT"       "AAAGCTC"      "TGGCATAA"     "TTGTATCCT"    "AAACTCGGCA"
[6] "ATGCGTGTCCT"  "CTCACGCGCCGG"
```

Even more simple...

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

```
[1] "TVNKP"        "DQQVFQ"       "KSLILRI"      "EMNMIEVC"     "MDRMAWTED"
[6] "DNYWFETTEK"   "CNFPRTVCMFI"  "QSDHGRQEIRDC"
```

Key-Point: Wrting 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.