

Class 6: R functions

Joseph Girgiss (PID: A17388247)

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 separated inputs to the function.
- The **body**, lines of R code that do the work of the function.

Our first function:

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

Let's test out 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 can be useful here to work with nucleotides A, C, T, G and return 3 of them.

```
sample(c("A", "T", "C", "G"), size=3)
```

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

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

```
[1] 1 5 2
```

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

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

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

```
generate_dna <- function(length) {
  n <- c("A", "T", "C", "G")
  v <- sample(n, size=length, replace = TRUE)
  cat("Well done you!\n")
  return(v)
}
```

```
generate_dna(30)
```

Well done you!

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

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

Well done you!

```
s
```

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

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

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

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

Well done you!

```
[1] "TTACGACTCG"
```

A more advanced example

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

```

generate_protein <- function(length, collapse = TRUE) {
  # Define the 20 standard amino acids
  amino_acids <- c("A", "R", "N", "D", "C", "E", "Q", "G", "H",
    "I", "L", "K", "M", "F", "P", "S", "T", "W",
    "Y", "V")
  
  # Generate a random sequence
  seq <- sample(amino_acids, size = length, replace = TRUE)
  
  # Return as string or vector depending on user input
  if (collapse) {
    return(paste(seq, collapse = ""))
  } else {
    return(seq)
  }
}

```

```

# Generate a 15-amino-acid protein sequence as a string
generate_protein(15)

```

```
[1] "GKERWEAIFVESMTE"
```

```

# Generate a 10-amino-acid protein sequence as a vector
generate_protein(10, collapse = FALSE)

```

```
[1] "H" "P" "P" "A" "Y" "E" "A" "I" "T" "M"
```

```
generate_protein()
```

```
[1] "GMIPNMKYYHHAWHAFLWMC"
```

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

One approach to do this is 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 values 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")
  cat(generate_protein(i))
  cat("\n")
}
```

```
> 5
NIWPH
> 6
PNILLP
> 7
LSDGFHW
> 8
ADIETSEE
> 9
DWMAFNEFL
> 10
TYFTKRKRKV
> 11
AYDTYICRRAC
> 12
WIHHRCHTEEQN
```

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

```
[1] "VVQGM"          "EHCHVP"         "NGPRYNI"        "NTPNCKWR"       "HRKGPRIPV"
[6] "GATHMVS AWM"   "EEHEQYM QWQY"  "HTQYQDEVFWI Q"
```

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