Class06: R Functions

Fan Wu(PID:A15127541)

All functions in R have at least 3 things:

- A **name**, we picked this and use it to call the function.
- Input **arguments**, there can be multiple comma seperated input 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,c(20,1130))
```

```
[1]   30 1140
```

```
add(10)
```

```
[1] 11
```

```
add(10, 10)
```

```
[1] 20
```

A second function

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

The `sample()` function could be useful

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

```
[1]  5  3 10
```

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

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

```
generate_dna<-function(size=10, nucleotides = c("A","C","G","T"), fasta = FALSE){
  # generate random choice of nucleotides
  v<- sample(n, size, replace = TRUE)

  # print a message to user
  cat("Well done you!\n")

  # Make a single element vector of string, rather than a vector of many elements of nucleoti
  s<- paste(v, collapse = "")

  # logic that help decide the form of output
  if(fasta){
    return(s)
  }else{
    return (v)
  }

}
```

```
generate_dna(3)
```

```
Well done you!

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

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

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

```
Well done you!

[1] "GAAGAGCTCA"
```

A more advanced example

Make a 3rd function that generates protein sequence of a use specified length and format.

```
#' Generate a random protein sequence
#'
#' @param length Integer. Number of amino acids in the sequence.
#' @param output_type Character. "string" for a single sequence string, "vector" for a charac
#'
#' @return A protein sequence as a string or vector, depending on output_type.
#' @examples
#' generate_protein(10, "string")
#' generate_protein(8, "vector")
generate_protein <- function(length, output_type = "string") {
  amino_acids <- c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I",
                   "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V")
  if (!is.numeric(length) || length < 1 || length != as.integer(length)) {
    stop("length must be a positive integer")
  }
  if (!is.character(output_type) || !(output_type %in% c("string", "vector"))) {
    stop("output_type must be a character string: 'string' or 'vector'")
  }
  seq <- sample(amino_acids, length, replace = TRUE)
  if (output_type == "string") {
```

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

Try this out..

```
generate_protein(10)
```

```
[1] "LWEVEVEHVI"
```

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

- 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

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

```
>6
DNLNPV
>7
GMCLPAH
>8
MRGSMNYQ
>9
DNHADQEWI
>10
FVGHRENYRD
>11
GYQALQLARGD
>12
STSILVHNCKNR
```

- A very useful 3rd R specific approach is to use the `sapply()` function

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

```
[1] "FRREI"        "KSYIVG"       "HCFFNGR"      "MFHYPWMM"     "LWKSVIQFV"
[6] "AFLASVEFIR"   "GEAWLQTRYRC"  "MWTCGFHDRFHR"
```

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