

R Functions Lab Class_06

Tessa Sterns PID:18482353

Every function in R contain at least 3 things:

- A **name** that we choose.
- Input **arguments**, multiple possible seperated by a comma.
- The **body**, lines of R code that do the work of the function.

My first lil function;

```
add <- function(x, y = 5) {x + y}  
add(10)
```

```
[1] 15
```

```
add(10,20)
```

```
[1] 30
```

```
add(c(1,2,3), 2)
```

```
[1] 3 4 5
```

The Second Function!

A sequence generator The `sample()` function can be useful in this

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

```
[1] 2 10 5
```

```
sample(c("A", "G", "T", "C"), size = 20, replace = TRUE)
```

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

```
DNA_Generator <- function( b=10 ) {  
  v <- sample(c("A", "G", "T", "C"), size = b, replace = TRUE)  
  cat("Good Job\n")  
  return(v)}  
DNA_Generator(30)
```

Good Job

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

```
DNA_Generator()
```

Good Job

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

```
s <- DNA_Generator(20)
```

Good Job

```
s
```

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

I want the option to return a single element vector “TATTAA” The `paste0()` with `collapse = “ ”` will reduce a vector down to a single element.

```

DNA_Generator2 <- function( b=10, type = c("multi", "fasta")) {
  type <- match.arg(type)
  v <- sample(c("A", "G", "T", "C"), size = b, replace = TRUE)

  ## single element vector
  cat("Good Job\n")
  if (type == "fasta") {
    return(paste0(v, collapse = ""))
  } else {
    # Return the vector of individual bases
    return(v)
  }
}
DNA_Generator2(40, "fasta")

```

Good Job

```
[1] "CGGGCGCCTCGGAGTTGCATTGGACCTAGTGCAACGTGTA"
```

Protein sequence generator

```

Protein_Gen <- function( b=10, type = c("PolyP", "multi"))
  ## make the default return the first argument in type()
  {
    type <- match.arg(type)

    AA <- sample(c("A","R", "N", "D", "Q", "H", "I",
                  "E", "G", "T", "C", "L", "K", "M",
                  "F", "P", "S", "W", "Y", "V"),
                size = b, replace = TRUE)

    ## single element vector

    if (type == "multi") {
      return(AA)
    } else {
      # Return the vector of individual bases
      return(paste0(AA, collapse = ""))
    }
  }
Protein_Gen(37, "multi")

```

```
[1] "T" "V" "E" "S" "Q" "A" "T" "I" "G" "Y" "K" "H" "F" "H" "V" "S" "K" "I" "H"  
[20] "M" "S" "G" "R" "Q" "R" "H" "V" "F" "R" "L" "I" "C" "V" "C" "R" "G" "L"
```

```
Protein_Gen(65)
```

```
[1] "KNEDDTWEDQTYKELTWSTSNQWNLCMDMINPMAKLPAWRGARVFTLQHENHQNWPNQKWILIWS"
```

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

```
sapply(5:12, Protein_Gen) |> cat(sep = "\n")
```

```
CWQTV  
EPEYLC  
RWVHAA  
CTLYSYIC  
NFSPGFADA  
GGWLSEGFTY  
DGAMYHTIWAI  
HEHQSIVVSMIL
```

Key-Point: writting functions in R is doable but not always intuitive. Starting with a working portion of code and using LLM tools to modify it can generate a more robust working function.