R Functions

Zainab Fatima (PID: A16880407)

Today we will get more exposure to functions in R. We call functions to do all our work and today we will learn how to write our own.

A First Silly Function (Writing Functions for Basic Math)

Note that arguments 2 and 3 have default values of 0 so we don't need to apply them when we call our function

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

```
add(1,1)
```

```
[1] 2
```

```
add (1, c(10, 100))
```

```
[1]  11 101
```

```
add(100)
```

```
[1] 100
```

```
#doesn't work since argument y is missing, "no default" error for y
#if you change y = 0 in the original argument of the add function, it will add to 100
#doesn't change if you provide a different value of y since y=0 is default if nothing else gi
```

```
add(100, 1, 1)
```

```
[1] 102
```

```
#error since there is an extra 1 that is an unused argument
#to make this work,you change the original function argument to default z = 0
```

A second more fun function

Let's write a function that generates random nucleotide sequences.

We can make use of the in-built `sample ()` function in R to help us here.

```
sample(x=1:10, size=9)
```

```
[1]  2  7  5 10  6  4  1  3  9
```

```
#randomly picks numbers between 1 and 10
#size = shows how many numbers you get back
```

```
#sample(x=1:10, size=11)
#gives an error since it ran out of numbers because it's not replacing the numbers back
```

```
sample(x=1:10, size=11, replace = TRUE)
```

```
 [1]  6  2 10  7  7  1  9  1  9  5  9
```

> Q. Can you use `sample()` to generate a random nucleotide sequence of length 5?

```
#save x as a vector
sample(x = c("A", "T", "C", "G"), size = 5, replace = TRUE)
```

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

> Q. Write a function `generate_DNA()` that makes a nucleotide sequence of a user specified length.

Every function in R has at least three things:

- a **name** (in our case "generate_DNA")
- one or more **input arguments** (the "length" of sequence we want)
- a **body** (R code that does the work)

```
generate_DNA <- function(length = 5) {
  bases <- c('A', 'T', 'C', 'G')
  sample(bases, size = length, replace = TRUE)
}
```

```
generate_DNA(10)
```

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

> Q. Can you generate a protein using `generate_protein()` function that returns amino acid sequence of a user requested length?

```
#install.packages("bio3d")
#Used to download amino acid codes
#biod::aa.table$aa1 to print out amino acids
aa <- bio3d::aa.table$aa1[1:20]
aa
```

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

```
generate_protein <- function (length = 5){
  sample (aa, size = length, replace = TRUE)
}
```

```
generate_protein(10)
```

```
 [1] "K" "A" "E" "I" "T" "G" "V" "W" "D" "M"
```

```
generate_protein (100)
```

```
  [1] "D" "I" "K" "Q" "S" "I" "I" "N" "I" "V" "M" "L" "T" "K" "T" "G" "H" "V"
 [19] "R" "A" "A" "W" "F" "W" "E" "L" "G" "V" "T" "I" "A" "L" "Q" "M" "T" "C"
 [37] "R" "P" "N" "T" "S" "G" "Q" "V" "Q" "I" "Y" "S" "T" "E" "E" "D" "Q" "D"
 [55] "Y" "E" "Y" "F" "C" "W" "N" "T" "M" "T" "T" "A" "S" "G" "K" "T" "G" "Y"
 [73] "W" "G" "L" "G" "M" "S" "F" "D" "I" "L" "M" "P" "T" "M" "W" "S" "C" "L"
 [91] "T" "H" "S" "N" "K" "G" "K" "E" "F" "S"
```

I want my output of this function not to be a vector with one amino acid per element but rather a one element single string.

```r
bases <- c("A", "G", "C", "T")
paste(bases, collapse = "")
```

```
[1] "AGCT"
```

Modifying generate_protein() function to have a string as a single element

```r
generate_protein <- function (length = 5){
  aa <- bio3d::aa.table$aa1[1:20]
  s <- sample (aa, size = length, replace = TRUE)
  paste(s, collapse ="")
}
```

```r
generate_protein (10)
```

```
[1] "IPIEQQHEVK"
```

> Q. Generate protein sequences from length 6 to 12

```r
generate_protein(length = 6)
```

```
[1] "RMETWW"
```

```r
generate_protein(length = 7)
```

```
[1] "KSTLGHM"
```

```r
generate_protein(length = 8)
```

```
[1] "QLISKNKV"
```

```r
generate_protein(length = 9)
```

```
[1] "DFHWQKKRY"
```

```r
generate_protein(length = 10)
```

```
[1] "GTHLMDSGFG"
```

```r
generate_protein(length = 11)
```

```
[1] "KQWKTMRNMLF"
```

```r
generate_protein(length = 12)
```

```
[1] "GANTSWQWCIDI"
```

```r
#very tedious to do this for every length
```

We can useful utility function ‘sapply()’ to help us “apply” our function over all the values 6 to 12.

```r
answer <- sapply(6:12, generate_protein)
answer
```

```
[1] "FMTWLW"        "EDDKLYL"       "LKESFVFF"      "CYPTPIRPR"     "ACIKRHQLWD"
[6] "THPFYQKFSDH"   "RPLNRHGFGVLE"
```

We want to test if these sequences are unique. We need to put it into FASTA format for BLAST.

```r
cat( paste(">ID.", 6:12, sep = "", "\n", answer, "\n"))
```

```
>ID.6
FMTWLW
 >ID.7
EDDKLYL
 >ID.8
LKESFVFF
 >ID.9
CYPTPIRPR
 >ID.10
ACIKRHQLWD
```

```
 >ID.11
THPFYQKFSDH
 >ID.12
RPLNRHGFGVLE
```

```
#\n creates new line
```

> Q. Are any of these sequences unique in nature - ie never found in nature? We can search "refseq-protein" and look for 100% identity and 100% coverage.

Sequence ID# 9, 10, 11, and 12 (lengths 9-12 aa) were unique in nature since they lack 100% identity and 100% coverage after the BLAST search. This makes sense because the higher the length, the narrower the options get for a full match to a protein in nature.