R Functions

Snehita Vallumchetla (PID: A16853399)

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

A first silly function

Can also give our functions a default argument, so if you do not give it a y or z-value, the function will use zero as a default.

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

In order to use the function, must first run the code cell containing it. R will take variables by order, it knows the first arugment is x and the second is y.

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

```
[1] 2
```

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

```
[1]  11 101
```

```
add(100)
```

```
[1] 100
```

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

```
[1] 102
```

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. This function can not return more values than the sample size, unless you specify the replace argument to TRUE, which will now allow replacement when sampling from a vector.

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

```
[1]  1  4  6  2  7  5  8 10  9
```

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

```
 [1] 1 5 4 3 1 9 1 8 4 5 9
```

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

```
nucleotides = c('A', 'C', 'T', 'G')

sample(nucleotides, size = 5, replace = TRUE)
```

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

> Q. Generate a funciton `generate_dna()` that makes a nucleotide sequence of a user specified length.

Every function in R has at least 3 things: * a **name** (in our case "generate_dna") * one or more **input arguments** * a **body** (that does the work)

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

Test function: DNA sequence of length 15

```
generate_dna(15)
```

```
[1] "TCTGGGGACAACGTG"
```

> Q. Can you write a `generate_protein()` function that returns amino acid sequences of a user specified length.

One way to get the amino acids is by installing the bio3d package!

```
#install.packages("bio3d")

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

```
generate_protein(6)
```

```
[1] "ASNGGE"
```

I want my output of this function not to be a vector with one amino acid per element but rather a single string, I can do this using the `paste(x, collapse = "")` function. I modified both the `generate_protein()` and `generate_dna()` functions above by adding the paste function within them to generate a single string output.

```
bases <- c('A', 'C', 'T', 'G')

paste(bases, collapse = '')
```

```
[1] "ACTG"
```

> Q. Generate protein sequences from length 6 to 12?

We can use an apply function such as `sapply()` to help us apply our function over all the values from 6 to 12.

```
ans <- sapply(X = 6:12, FUN = generate_protein)
```

> Q. Has nature invented these sequences? (Get them into the fasta format in order to paste them into BLAST, >ID.1—)

```
cat(paste('>ID', 6:12, sep = '', '\n', ans,'\n'), sep = '')
```

```
>ID6
PLHPSF
>ID7
HRRRRNA
>ID8
TLSYINDM
>ID9
IVLLKESEH
>ID10
CYYWSDPNVD
>ID11
HMGNYHCPMTW
>ID12
GRPEFCLNHGYH
```

> Q. Are any of these sequences unique in nature - i.e. never found in nature. We can search "refseq-protein" and look for 100% Ide and 100% coverage matches using FASTAp.

My amino acid sequence with the length of 6, 7, 8, all have full coverage and 100% Ide. My protein sequence with length 9 has 100% Ide but only 89% coverage. My sequence with length 10 has 88% Ide and 90% coverage. My sequence with length 11 has 100% Ide, but only 89% coverage. My sequence with length 12 has 83% Ide with 92% coverage. The reason why the longer sequences are not found in nature are because the randomization increases with increasing amount of amino acids that are randomized in the sequence!