R functions

Lily Huynh (PID: A16929651)

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

Note that argument 2 and 3 have default values (because we set y=0 and z=0) so we don't have to supply them when we call our function.

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

Can I just use this?

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

```
[1] 2
```

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

```
[1]  11 101
```

```
add(100)
```

```
[1] 100
```

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

```
[1] 111
```

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] 10  1  3  5  6  9  4  8  7
```

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

```
 [1] 10  6  5  7 10  1  8  3  1  8  5
```

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

```
sample (x= c( "A", "T", "C","G"), size = 5, replace = TRUE)
```

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

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

Every function in R has at least 3 things:

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

```
generate_dna <- function (length =5) {
  bases <- c ("A", "T", "C","G")
  sample(x= bases, size = length, replace = TRUE)}
```

```
generate_dna(10)
```

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

```
generate_dna(100)
```

```
  [1] "C" "A" "C" "C" "T" "G" "A" "C" "A" "A" "A" "T" "G" "G" "C" "A" "A" "T"
 [19] "T" "G" "A" "G" "G" "C" "C" "A" "A" "T" "A" "A" "C" "T" "G" "T" "A" "C"
 [37] "T" "T" "A" "C" "G" "C" "G" "G" "G" "G" "C" "G" "G" "T" "T" "G" "T" "T"
 [55] "T" "G" "G" "G" "A" "A" "G" "A" "A" "T" "C" "C" "A" "G" "G" "G" "A" "G"
 [73] "C" "C" "C" "G" "T" "T" "A" "T" "A" "A" "C" "T" "T" "G" "G" "A" "C" "C"
 [91] "G" "A" "T" "A" "G" "T" "A" "C" "T" "A"
```

Q. Can you write a `generate_protein` function that returns amino acid sequence of a use requested length?

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

```
generate_protein (6)
```

```
[1] "Q" "E" "I" "R" "E" "C"
```

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.

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

```
[1] "ACGT"
```

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

```
generate_protein()
```

```
[1] "QFIHH"
```

Q. Generate protein sequences from length 6 to 12?

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

```
[1] "QYEMGL"
```

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

```
[1] "LHSKKKR"
```

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

```
[1] "SVVNFLSP"
```

We can use the useful utility function `sapply()` to help us "apply" our function over all the values 6 to 12

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

```
[1] "NEMGTT"       "ASEKLHY"      "IQYVKRLG"     "AGMGTHDMD"    "TQKPNWMVYG"
[6] "MNGSQSVRWQN"  "GFYYTDTTEKWP"
```

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

```
>ID.6
NEMGTT
 >ID.7
ASEKLHY
 >ID.8
IQYVKRLG
 >ID.9
AGMGTHDMD
 >ID.10
TQKPNWMVYG
 >ID.11
MNGSQSVRWQN
 >ID.12
GFYYTDTTEKWP
```

Q. Are any of these sequences unique in nature - i.e. never found in nature. We can search “refseq-protein” and look for 100% Identity and 100% coverage matches with BLASTp

All of these sequences are found in nature, due to majority of the matches from BLASTp having 100% identity and 100% coverage.