R Functions Class 06

Dylan Mullaney (A16869792)

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.

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

add (5, 6)
```

```
[1] 11
```

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

```
[1]  11 101
```

Make sure to send it to the brain if you add a new function

```
add (100)
```

```
[1] 100
```

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

add (8, 17, 45)
```

```
[1] 70
```

Note that arguments can have default values, which allows it to run with less outputs than there are variables.

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]  6  9  8  3  2  4  1  5 10
```

Randomly repeating, but not reusing the same number

```
#{r}    #sample(x=1:10, size=11)
```

Produces error, doesn't work because it's asking for too many outputs and there's not enough option to not repeat one

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

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

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

- Use "replace=TRUE" to be able to repeat values when you're asking for more outputs than you have options.
- Using the small "c" makes your values into a vector

> Q. Write a function called 'generate_dna()' that makes a nucleotide sequence of a user specified length.

Every function in R has at least 3 things:

- a **name** (ie. generate_dna())
- one or more **input arguments** (ie. length of sequence)
- a **body** (what does the work)

```
generate_dna <- sample (x = c("A", "G", "T", "C"), size= 12, replace = T)
generate_dna
```

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

Let's try a way to personalize the length each time.

```
generate_dna2 <- function(length=12) {
  bases <- c ("A", "C", "G", "T")
  sample(bases, size = length, replace = TRUE)
}
```

```
generate_dna2(10)
```

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

```
generate_dna2(100)
```

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

Can you write a 'generate_protein()' function that returns amino acid sequences of a user requested length?

```
##First, pull in amino acids list from bio3d. 1st line is whole library, 2nd line is more sp

library(bio3d)

bio3d::aa.table$aa1 [1:20]
```

```
 [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) {
  AA= c (bio3d::aa.table$aa1 [1:20])
  sample(AA, size=length, replace=TRUE)
}

generate_protein(12)
```

```
 [1] "N" "H" "K" "A" "K" "Y" "S" "A" "N" "D" "A" "D"
```

I want my output of this function not to be a vectore with one amino acid per element but rather a one element single string (ie like a word with all the letters pasted together). The paste() function puts things together; paste (x, collapse=" "). Collapse tells what you want in between the values.

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

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

```
paste(generate_protein(), collapse="")
```

```
[1] "EQTMNWDLQKHQDHQGWPLC"
```

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

```
  generate_protein()
```

```
[1] "CYHMC"
```

Q. Generate protein sequences from length 6 to 12

We can use the useful utility function 'sapply()' to help us "apply" our function over all the values from 6-12. Want to know more about it? Use ?sapply

```
sapply(6:12, generate_protein)
```

```
[1] "LLILIN"       "YAGNQYK"      "NVRSTNCR"     "NCTVINNEP"    "FCERAMYTYI"
[6] "CIECHHCKLGS"  "RRWQNTTPGLIP"
```

I want teh sequences to be labeled more nicely.

```
paste(">ID", 6:12)
```

```
[1] ">ID 6"  ">ID 7"  ">ID 8"  ">ID 9"  ">ID 10" ">ID 11" ">ID 12"
```

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

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

```
>ID.6
HSNYMG
>ID.7
IDLPAEN
>ID.8
KYYDCFDP
>ID.9
WHTTCPDDY
>ID.10
KQSHERMLKP
>ID.11
GFYFYQVYLHL
>ID.12
ADMNMRWADLYT
```

> 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 with blastP.

Some sequences did have 100% identity over 100% query coverage. However, there are so many possible amino acid combinations over these varying lengths that it makes sense that there were 4 of the 7 tested sequences that did not have an exact match.