R Functions

Jordan Prych (PID: A17080226)

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 Little Function

Let's write a function that does some basic math. Note that arguments 2 and 3 have default values (because we set y=0 and z=0, we do not have to supply them when we call our function)

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

Can I just use this? No, the function needs to be run first.

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

```
[1] 2
```

```
add(x=1, y=c(10, 100))
```

```
[1]  11 101
```

```
add(100)
```

```
[1] 100
```

If you only input one value into add(), this returns an error because the function shows that there are two arguments, but this code is missing "y" because there is no default. Add a default value to y(y=0) in function.

If you were to code add(100, 10, 1), this would output an error because there are three values, and the third value is not defined in the function. To add three values, create another input "z".

A Second More Fun Function

Let's write a function that generates random nucleotides.

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

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

```
[1] 4
```

This does not "replace" the values after it randomly picks one, so you cannot pick more than the size of the population. Use the argument`replace` set to TRUE to replace values and choose more than the population size.

```
sample(x=1:10, size=20, replace=TRUE)
```

```
 [1] 10  6  4  9  3  1 10  6  4  2 10  8  1 10  9  3  6  1  8  7
```

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

```
x<- c("A", "G", "T", "C")

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

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

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

Every function in R has at least three things:

1. a **name** (in our case "generate_dna")
2. one or more **input arguments** (the "length" of sequence we want)

3. a **body** (that does the work)

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

```
generate_dna(3)
```

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

> Q. Can you write a `generate_protein()` function that returns amino acid sequence of a user requested legnth?

Install bio3d package using `install.packages()` to access amino acid sequences.

```
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=2) {
  aa <- bio3d::aa.table$aa1[1:20]
  sample(aa, size=length, replace=TRUE)
}
```

```
generate_protein(18)
```

```
 [1] "S" "K" "S" "M" "P" "N" "M" "K" "E" "G" "S" "S" "G" "Y" "F" "I" "D" "P"
```

I want my output of this function not to be a vector with one amino acid per element, but rather as one element.To have all values printed together, use `paste()`. Second input can either be collapse=" " or separate= " ".

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

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

```
generate_protein <- function(length=2) {
  aa <- bio3d::aa.table$aa1[1:20]
  s <- sample(aa, size=length, replace=TRUE)
  paste(s, collapse="")
}
```

```
generate_protein(13)
```

```
[1] "SICYVDMLPTRIK"
```

Q. Generate proetin sequences from length 6 to 12.

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

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

```
[1] "PDWITT"       "DYIMEEG"      "YFSIHPFW"     "KHFVTIKVY"    "EQRHHSCKRQ"
[6] "KPKSSQHQCES"  "QQVVDAWVDLVA"
```

FASTA format: >ID.1_____ You can use `paste()` to generate sequences in FASTA format

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

```
>ID.6
PDWITT
 >ID.7
DYIMEEG
 >ID.8
YFSIHPFW
 >ID.9
KHFVTIKVY
 >ID.10
EQRHHSCKRQ
 >ID.11
KPKSSQHQCES
 >ID.12
QQVVDAWVDLVA
```

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 mtached with BLASTp.

Some sequences of shorter length may be found in the BLASTp, but as the sequence length increases, there is a lesser chance that this protein exists in nature since the possibilities of amino acid sequences are astronomical.