Class 6: R functions

Carly Chang (PID A16843962)

Today, we will get more exposure to functions in R. We can call functions to do all our work and today we will learn how to write our own using `function()`.

First function

Argument 2 has default value (y set to 0) so we don't have to supply them when we call our function.

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

Can I just use this? You must run the code chunk with the function first.

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

```
[1] 2
```

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

```
[1]  11 101
```

```r
add(100) #This will not work if y argument is missing with no default in function
```

```
[1] 100
```

```
#add(100,10,1) #This does not work. Arguments must match
```

Second 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.

```
sample(x=1:10, size=9) #default is no replacement (aka no repeats) so size cannot be larger
```

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

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

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

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

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

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

> Q. Write a function `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** (ie. the length of sequence we want)
- a **body** (that does the work)

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

```
generate_dna(10)
```

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

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

Install bio3d for this.

```
aa <- bio3d::aa.table$aa1[1:20] #gives 20 naturally occurring amino acids
```

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

```
generate_protein(8)
```

```
[1] "L" "D" "C" "D" "G" "H" "R" "L"
```

I want my output of this function not to be a vector with one amino acid per element, but rather a single string. I will achieve this by using paste(x,collapse=" ")

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

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

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

```
generate_protein(8)
```

```
[1] "AFVFFGMM"
```

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

```
#generate_protein(6:12) #length is not vectorized in the function so you cannot put a vector
```

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

```
ans <- sapply(6:12, generate_protein) #applies 6 to 12 to function generate_protein
ans
```

```
[1] "KDYTPL"       "SCMVAEH"      "TITEDSAK"     "AWVGKDSWP"    "EWVFPYDMWS"
[6] "DMSNIYYRSWA"  "VYQPHTGMKICA"
```

Put in FASTA format:

```
cat(paste(">ID", 6:12, sep="", "\n", ans, "\n")) #"\n" is new line and cat() outputs "\n" as
```

```
>ID6
KDYTPL
 >ID7
SCMVAEH
 >ID8
TITEDSAK
 >ID9
AWVGKDSWP
 >ID10
EWVFPYDMWS
 >ID11
DMSNIYYRSWA
 >ID12
VYQPHTGMKICA
```

> 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.

ID7, ID9, ID10, ID11, ID12 have no matches. Randomized shorter sequences will have higher matches while longer sequences will be less likely to have matches.