Lab 6: R Functions

Emily Ignatoff (A16732102)

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

##A first silly function

Note that the second and third arguments have a set default of 0. This prevents us from needing to supply those variable each time.

```
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
```

> Note: alway remember to run your function definition before you attempt to use the function! Also, the number of entries must match the number of arguments, or be less than as long as defaults are set to 0

```
add(100)
```

```
[1] 100
```

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

```
[1] 111
```

##A second more fun function:

Let's write a function that generates random nucleotide sequences!

We can use the built-in `sample()` function to achieve this.

```r
#This function randomly picks numbers between 1 and 10 without repeating any values
#(default replace=FALSE)
sample(x=1:10, size=5)
```

```
[1]  8  3  9 10  7
```

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

```
 [1] 10  8 10  5  1  9 10  6 10  1  9
```

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

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

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

> Q: Write a function that will complete this for a user specified nucleotide length:

Every function in R has at least three things:

- a **name** (in our case "generate_dna")
- one or more **input arguments** (user designated "length")
- a **body** (does the work)

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

```r
generate_dna(15)
```

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

> Q. Can yoou write a `generate_protein()` function to return an amino acid sequence of user designated length?

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

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

```r
generate_protein(6)
```

```
[1] "P" "C" "M" "D" "E" "S"
```

I want my output of this function to have no quotes, and rather be a continuous string as opposed to individual elements

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

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

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

generate_protein(6)
```

```
[1] "ADKISQ"
```

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

We can use the `sapply()` utility function to accomplish this by applying our function over 6 to 12:

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

```
[1] "LKSGIA"       "MWVPTIE"      "DWQFCQVW"     "QYDKGYIGS"    "EWDHTKLTSL"
[6] "SQWPRNCEKDG"  "RQNNPCNKINAE"
```

Let us now format these sequences in fasta format such that they could be used in BLAST.

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

```
>ID_6
LKSGIA
 >ID_7
MWVPTIE
 >ID_8
DWQFCQVW
 >ID_9
QYDKGYIGS
 >ID_10
EWDHTKLTSL
 >ID_11
SQWPRNCEKDG
 >ID_12
RQNNPCNKINAE
```

> Q. Are any of these sequences unique in nature or do they have a match in BLAST? We will search "refseq-protein" and look for 100% coverage and 100% ID matches with BLASTp.

For shorter sequences (6-8 aa long) we can find exact matches in nature. Once we reach 9+ amino acids the sequences are more frequently unique to nature due to the high number of amino acid permutations.