R functions

Ashley Allen (PID: A14633373)

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 arguments 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  8  9  4  2  6  3  1  7
```

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

```
 [1]  7  3  9  7  2 10  6 10  8  8  5
```

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

We use c here to make a vector.

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

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

> Q. Write a function `generate_dna()` that makes a nucleotide sequence of a user specificed 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 sequence we want)
- a **body** (R code that does the work)

Here we assign a default length of 5 that we can overide later.

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

```
generate_dna(10)
```

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

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

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

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

```
generate_protein(10)
```

```
 [1] "C" "R" "Q" "T" "P" "P" "F" "D" "P" "L"
```

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 (like a word).

```
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(10)
```

```
[1] "NGRCEVKFNN"
```

Q. Generate protein sequences from length 6-12?

This technically works, but its NOT efficient.

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

```
[1] "CQLNIY"
```

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

```
[1] "FAKRCCF"
```

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

```
[1] "RAWYHGPA"
```

We can use the useful utility function `sapply()` to help us "apply" our function over all the values 6-12. (x = the values we want it to return, FUN = function)

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

```
[1] "FFCRGM"       "SHHGMHF"      "HDHVTLME"     "NVCLEIQLL"    "GSCFARIILA"
[6] "FCINGFWKWKT"  "QFTESNHVCLND"
```

Remember that FASTA format is >ID.1..... We can use the paste function to insert the `cat()` function to print it out in respect to a new line or tab. The order matters.

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

```
>ID.6
FFCRGM
 >ID.7
SHHGMHF
 >ID.8
HDHVTLME
 >ID.9
NVCLEIQLL
 >ID.10
GSCFARIILA
 >ID.11
FCINGFWKWKT
 >ID.12
QFTESNHVCLND
```

Now we want to see if these are unique. BLAST uses a sliding window of 9 characters.

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

Paste our sequences into protein BLAST and select “refseq-protein.”

There are unique sequences from 8:12. We see a drop in coverage and % identity.