Class 6: R Functions

AUTHOR
Claire Lua A16922295

1. Function Basics

Let’s start writting our first silly function to add some numbers:

Every R function has 3 things: - name (we get to pick this) - input arguments (there can be loads of these separated by a comma) - the body (the R code that does the work)

```r
add <- function(x, y=100, z=0) {
  x + y + z
}
```

I can just use this function like any other function as long as R knows about it (i.e. run the code chunk)

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

```
[1] 101
```

```r
add(x=c(1,2,3,4), y=100)
```

```
[1] 101 102 103 104
```

```r
add(1)
```

```
[1] 101
```

Functions can have “required” input arguments and “optional” input arguments. The optional arguments are defined with an equals default value (`y=0`) in the function definition

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

```
[1] 111
```

2. Generate DNA Function

> Q. Write a function to return a DNA sequence of a user specified length. Call it `generate_dna()`

The `sample()` function can help here

```r
generate_dna <- function(size=5) {
}

students <- c("jeff", "jeremy", "peter")

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

```
[1] "jeff"  "peter" "peter" "jeff"  "peter"
```

Now work with `bases` rather than `students`

```r
bases <- c("A", "C", "G", "T")
sample(bases, 10, TRUE)
```

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

Now I have a working `snippet` of code, I can use this as the body of my first function version here:

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

```r
generate_dna(100)
```

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

I want the ability to return a sequence like "AGTACCTG" i.e. a one element vector where the bases are all together.

```r
generate_dna <- function(size=5, together=TRUE) {
  bases <- c("A", "C", "G", "T")
  sequence <- sample(bases, size, TRUE)
  if(together){
    sequence <- paste(sequence, collapse="")
  }
  return(sequence)
}
```

```r
generate_dna(100)
```

```
[1]
"TAAGTATGTGGGGTTCAATTGACATTTTGAGTCCCCCGAGGCAAGATCCCGGGATCTAGCCTGTCCGCTTCTAGGGCTCTCATGT
GGTTCAATGGTATTG"
```

3. Generate Protein Function

> Q. Write a protein sequence generating function that will return sequences of a user specified length

We can get the set of 20 natural amino-acids from the **bio3d** package.

```
generate_protein <- function(size=6, together=TRUE){
  ##Get the 20 amino acid
  aa <- bio3d::aa.table$aa1[1:20]
  sequence <- sample(aa, size, TRUE)

  ##Optionally return a single element string
  if(together){
    sequence <- paste(sequence, collapse="")
  }
  return(sequence)
}
```

```
generate_protein(10)
```

```
[1] "SFEFKRHEGF"
```

> Q. Generate random protein sequences of length 6 to 12 amino acids

```
##generate_protein(6:12)
```

We can fix this inability to generate multiple sequences by either editing and adding to the function body code (e.g. a for loop) or by using the R **apply** family of utility functions

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

It would be cool and useful if I could get FASTA format output

```
id.line <- paste(">ID.", 6:12, sep="")
seq.line <- paste(id.line, ans, sep="\n")
cat(seq.line, sep="\n")
```

```
>ID.6
VSELSD
>ID.7
MNYDRWI
>ID.8
QCATLSCT
>ID.9
AKWQPDRWQ
>ID.10
YGCGECKREA
>ID.11
NEITRAAHVSK
```

```
>ID.12
VMVVTTFRMNRD
```

> Q. Determine if these sequences can be found in nature or are they unique? Why or why not?

I BLASTp searched my FASTA format sequences against NR and found that length 6, 7, and 8 are not unique and can be found in the databases with 100% coverage and 100% identity.

Random sequences of length 9 and above are unique and can’t be found in the databases.