Class 6: R Functions

Hyejeong Choi (PID: A16837133)

Table of contents

1. Generating functions

Let's start writing 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)

```
add <- function(x, y=10, 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)

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

```
[1] 101
```

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

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

```
add(1)
```

```
[1] 11
```

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.

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

```
[1] 111
```

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

The `sample()` function can help here

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

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

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

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

2. Generate DNA sequence

Now work with `bases` rather than `students`

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

sample(bases, size = 10, replace = TRUE)
```

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

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

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

```
generate_dna()
```

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

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

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

  if(together){
    sequence <- paste(sequence, collapse = "")}

  return(sequence)}
```

```
generate_dna()
```

```
[1] "GGTCT"
```

3. Generate protein sequence

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

```
generate_protein <- function(length=10, together=TRUE){

  ## get the 20 amino acids as a vector
  amino_acids <- bio3d::aa.table$aa1[1:20]
  sequence_protein <- sample(amino_acids, size=length, replace=TRUE)

  ## optionally return a single element string
  if(together){
    sequence_protein <- paste(sequence_protein, collapse="")}

  return(sequence_protein)}
```

```
generate_protein()
```

```
[1] "QDRWMHFINN"
```

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

```
generate_protein()
```

```
[1] "GIMRHQLGSC"
```

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.

```
sapply(6:12, generate_protein)
```

```
[1] "MYGYIH"       "QFDILVH"      "ANRQDSHV"     "AVTDWPEEW"    "HALFMEDNNC"
[6] "RSSYCDGKKGL"  "FTREPPNTEWAG"
```

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

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

```
[1] "KAYYCH"       "QVIIGHD"      "NDAMDHHQ"     "TMLMDIYIH"    "AQIGQEDVAR"
[6] "YAFAVMWAMLI"  "IDCPQFNYFARS"
```

```
cat(ans, sep="\n")
```

```
KAYYCH
QVIIGHD
NDAMDHHQ
TMLMDIYIH
AQIGQEDVAR
YAFAVMWAMLI
IDCPQFNYFARS
```

I want this to look like:

ID.6 KIWHND ID.7 FENRCVK ID.8 MSKHTDFI

The functions `paste()` and `cat()` can help us here

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

```
> ID.6
KAYYCH
> ID.7
QVIIGHD
> ID.8
NDAMDHHQ
> ID.9
TMLMDIYIH
> ID.10
AQIGQEDVAR
> ID.11
YAFAVMWAMLI
> ID.12
IDCPQFNYFARS
```

```
id.line <- paste("> ID.", 6:12, sep="")
id.line
```

```
[1] "> ID.6"  "> ID.7"  "> ID.8"  "> ID.9"  "> ID.10" "> ID.11" "> ID.12"
```

```
seq.line <- paste(id.line, ans, sep="\n")
cat(seq.line, sep="\n", file="myseq.fa")
```

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 the refseq_protein database and found that length 6, 7, and 8 are not unique and can be found in the database with 100% coverage and 100% identity. Length 9, 10, 11, and 12 are unique and can't be found in the database.

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

```
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"
```