Class 6: R functions

Leah Kim (A16973745)

Table of contents

1. 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

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

```
[1] 101
```

```
add(1)
```

```
[1] 11
```

Functions can either have "required" input arguments and "optional" input arguments. The optional arguments are defined with an equals default value (`y = 10`) 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()`

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

students <- c("jeff", "jeremy", "peter")
sample(students, size = 1)
```

```
[1] "jeremy"
```

```r
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, size = 10, replace = TRUE)
```

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

Now I have a working 'snippet' of code I can use.

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

```r
generate_dna(100)
```

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

```r
generate_dna()
```

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

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 = size, replace = TRUE)
  if(together) {
    sequence <- paste(sequence, collapse = "")
  }
  return(sequence)
}
```

3. Generate Protein Function

We can ge thte set of 20 natural animo-acids from the **bio3d** package

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

```r
generate_protein <- function(size = 5, together = TRUE) {
  #get the 20 amino acids as a vector
  aa <- bio3d::aa.table$aa1[1:20]
  sequence <- sample(aa, size = size, replace = TRUE)

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

Q. Generate random proteins equences of length 6-12 amino acids.

```r
#generate_protein(6:12) returns an error
```

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

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

```
[1] "DYKFYI"        "TRKMRSN"       "PFHQESGI"      "EWPLWAPDI"     "MLGRHYRRPW"
[6] "WGNATTWFRTV"   "AYPKWFWLYITV"
```

It would be cool and useful if I could get FASTA format output. I want this to look like

```
>ID.6
HLDVLV
>ID.7
VREAIQN
>ID.8
WPRSKACN
```

The functions `cat` and `paste` can help us here.

```r
ans <- sapply(6:12, generate_protein)
cat(paste(">ID.",6:12, sep="", "\n", ans), sep ="\n")
```

```
>ID.6
LYMSWP
>ID.7
CYHYSCK
>ID.8
TLYCIPGW
>ID.9
FCDMILALV
```

```
>ID.10
HWQAQRDGLG
>ID.11
LCTGCAMYMYW
>ID.12
TDLAHPYEMWGS
```

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

The sequences generated are as followes:

```
>ID.6
SFDRHS
>ID.7
HQNLFYY
>ID.8
DSMEMNDL
>ID.9
STYFCEKGC
>ID.10
CVDIIEFNKR
>ID.11
ECFMCPHRVDN
>ID.12
PSRKPESIFEHE
```

I BlastP my FASTA format sequences against NR and found that the sequences of lengths 6 through 8 are not unique and found in the databases with 100% coverage and identity.

Random sequences of length 9 and above are unique and cannot be foud in the databases.