Class 6: R Functions

Jacqueline Cheung(A17085191)

2025-01-23

Today we will gain more exposure to functions in R. We call on functions to do all our work and today we will learn how to write our own.

A first silly function (writing functions for basic math)

Note that arguments 2 and 3 have default values of 0 so we don't need to apply them when we call our function

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

Can I just use this?

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

```
[1] 2
```

```r
add(x=1, y=c(10,100))
```

```
[1]  11 101
```

```r
add(100)
```

```
[1] 100
```

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

```
[1] 110
```

A second more fun function

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

We can make use of the in-built `samples()` function in R to help us here.

```r
sample(x=1:10, size=9)
```

```
[1] 10  8  3  1  4  7  9  6  2
```

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

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

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

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

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

> Q. Write a function `generate_dna()` that makes a nucleotide sequence of a user specified 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** (that does the work)

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

```r
generate_dna(10)
```

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

```r
generate_dna(100)
```

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

Q. Can you write a `generate_protein()` function that returns amino acids

```r
# install.packages("bio3d")

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

```r
generate_protein(10)
```

```
 [1] "P" "T" "A" "N" "E" "F" "A" "C" "E" "E"
```

I want my output of this function not to be a vector of one amino acid per element but rather a one element single string

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

```
[1] "ATGC"
```

```r
generate_protein=function(length=5){
  aa <-bio3d::aa.table$aa1[1:20]
  s<- sample(aa, size=length, replace=T)
  paste(s, collapse="")
}
```

```r
generate_protein(15)
```

```
[1] "FSNCVPIEKAFLGMT"
```

> Q. Generate protein sequences from lengths 6 to 12?

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

```
[1] "IWWPFM"
```

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

```
[1] "WESNLNC"
```

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

```
[1] "FNVYRVGQ"
```

We can use the useful utility function `sapply()` to help us "apply" our function over all the values

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

```
[1] "RHLNEE"       "GGQMVGG"      "WAQGPAKP"     "ICTSYFSHR"    "DNLHTNYSFG"  
[6] "PKHFPHRYKSA"  "TWQFMIMCDDGH"
```

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

```
>ID.6
RHLNEE
>ID.7
GGQMVGG
>ID.8
WAQGPAKP
>ID.9
ICTSYFSHR
>ID.10
```

```
DNLHTNYSFG
>ID.11
PKHFPHRYKSA
>ID.12
TWQFMIMCDDGH
```

> Q. Are any of these sequences unique in nature - i.e. never found in nature? We can search “ref-protein” and look for 100% Ide and 100% query cover

No. The first sequence had 3 hits ranging from 27%-33% identity however, no other sequence had any significant hits.