R Functions Lab

Amanda Wilpitz | PID: A17463962

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).

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

Can I just use this?

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

```
[1] 2
```

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

```
[1]  11 101
```

First I need to run the code chunk creating the function, before I can use it.

```r
add(100)
```

```
[1] 100
```

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

```
[1] 102
```

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.

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

```
[1]  4  8  2  9  5  6  3  1 10
```

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

```
 [1]  2 10  1  8  7 10  7  7 10  7  7
```

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

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

```
[1] "G" "G" "C" "G" "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("C", "G", "T", "A")
  sample(bases, size = length, replace = TRUE)}

generate_dna(length=12)
```

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

```r
generate_dna(10)
```

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

```
generate_dna(100)
```

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

> Q. Can you write a `generate _protein()` function that returns amino acid sequences of a user requested length?

Install `bio3d` package before running this.

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

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

```
generate_protein()
```

```
[1] "AGVEE"
```

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.

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

paste(bases, collapse = "")
```

```
[1] "AGCT"
```

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

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

```
[1] "CCLKYR"
```

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

```
[1] "RKDHHIC"
```

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

```
[1] "VQKTYWCN"
```

We can use the useful utility function `sapply()` to help us “apply” our function over all the values 6 to 12.

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

```
[1] "RAPLTD"       "SWIAVNM"      "VSNGQVEI"     "RHCACESLN"    "QGRMWGYFSF"
[6] "EKQHDSCVVVL"  "SSAHYAFPFWNG"
```

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

```
>ID6
RAPLTD
 >ID7
SWIAVNM
 >ID8
VSNGQVEI
 >ID9
RHCACESLN
 >ID10
QGRMWGYFSF
 >ID11
EKQHDSCVVVL
 >ID12
SSAHYAFPFWNG
```

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

Yes, ID6 and ID8 had 100% Ide and 100% coverage matches.

ID7, ID9, ID10, ID11, ID12 did not have a 100% Ide and 100% coverage match.

Randomized shorter sequences has higher changes to have matches while longer sequences will be less likely to have matches.