Class 06: R Functions

Aadhya Tripathi (PID: A17878439)

Table of contents

Background

Functions are at the heart of using R. Everything we do involves calling functions (including data input, analysis, results output).

All functions in R have at least 3 things:

1. A **name,** the thing used to call the function
2. 1+ input **arguments,** comma separated
3. The **body,** lines of code between curly brackets { } that do the work of the function

A first function

Write a silly, short function to add some numbers.

```
add <- function(x) {
  x + 1
}
```

Let's try the function!

```
add(100)
```

```
[1] 101
```

Will this work?

```
add( c(100,200,300) )
```

```
[1] 101 201 301
```

Modify to be more useful and add more than just 1

```
add <- function(x, y=1) {
  x + y
}
```

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

```
[1] 110
```

Will this work? Yes - if we assign a default value to y

```
add(100)
```

```
[1] 101
```

> **N.B.** Input arguments can be **required** OR **optional**. The optional values have a default fall-back, specified in the function code with an equals sign!

```
# add(100,200,300)
```

A second function

All functions in R look like this:

```
name <- function(arg) {
  body
}
```

The `sample()` function in R takes a random sample of a given size from the elements of a vector.

```
sample(1:10, size=4)
```

```
[1] 5 6 8 3
```

Q. Return 12 numbers picked randomly from the imput 1:10

```
sample(1:10, size=12, replace = TRUE)
```

```
 [1]  5  5  2 10  2  5  3  1  7  2  6  9
```

Q. Write the code to generate a random 12 nucleotide long DNA sequence

```
bases <- c("A", "C", "G", "T")
sample(bases, size=12, replace = TRUE)
```

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

Q. Write a first version funtion called `generate_dna()` that generates a user specified length `n` random DNA sequence.

```
generate_dna <- function(n=6) {
  bases <- c("A", "C", "G", "T")
  sample(bases, size=n, replace = TRUE)
}
```

Test:

```
generate_dna(20)
```

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

Q. Modify your function to return a FASTA like sequence (rather than an R vector of individual bases)

```
generate_dna <- function(n=6) {
  bases <- c("A", "C", "G", "T")
  paste(sample(bases, size=n, replace = TRUE), collapse = "")
}
```

Test:

```
generate_dna(10)
```

```
[1] "GTTACTGGTT"
```

> Q. Give the user the option to trurn output in FASTA format or standard multi-element vector.

```
generate_dna <- function(n=6, fasta = TRUE) {
  bases <- c("A", "C", "G", "T")
  seq <- sample(bases, size=n, replace = TRUE)

  if(fasta) {
  seq <- paste(seq, collapse = "")
  }

  return(seq)
}
```

Test:

```
generate_dna(n=8, fasta=FALSE)
```

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

```
generate_dna(n=20)
```

```
[1] "AGGTTGAGGGTCAACGGGCC"
```

A cool new function

> Q. Write a fucntion called `generate_protein()` that generates a user specified length protein sequence in FASTA format.

```
generate_protein <- function(n=6) {
  aa <- c("A","R","N","D","C",
          "E","Q","G","H","I",
          "L","K","M","F","P",
          "S","T","W","Y","V")
  seq <- sample(aa, size=n, replace = TRUE)
  return (paste(seq, collapse = ""))
}
```

Test:

```
generate_protein(12)
```

```
[1] "VGDICCQKRQNC"
```

> Q. Use your new `generate_protein()` function to generate sequences between length 6 and 12 amino-acids in length. Check if any of these are unique in nature (i.e. found in the NR database at NCBI).

```
for(i in 6:12) {
  cat(">seq", i, sep="", "\n")
  cat(generate_protein(i), "\n")
}
```

```
>seq6
FENHGA
>seq7
DCQMDSM
>seq8
SQTNVEIF
>seq9
LSPFPDMVG
>seq10
WGHPFCPMVV
>seq11
PAERFFCMKFC
>seq12
NAATMTNHSKVY
```