Class 06: R Functions

Mitchell Sullivan (PID: A18595276)

Table of contents

Background

Functions are at the heart of using R. Everything we do involves calling and using functions (from data input and analysis to outputs and results).

All functions in R have at least three things:

1. A **name** (the thing we use to call the function)
2. One or more input **arguments** that are comma separated
3. The **body** (lines of code between curly brackets { } that does the work of the function)

A first function

Let's write a silly wee function to add some numbers:

```
add1 <- function(x) {
  x + 1
}
```

Let's try it out:

```
add1(100)
```

```
[1] 101
```

```
#Will this work?
add1(c(100, 200, 300))
```

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

Yes!

Modify it to be more useful and add more than just 1

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

```
#Will this work?
add(100, 10)
```

```
[1] 110
```

Yes!

```
#Will this work?
add(100)
```

```
Error in add(100): argument "y" is missing, with no default
```

No!

Let's fix that error so that it will automatically add 1 when no second argument is passed

```
add <- function(x, y = 1) {
  x + y
}
add(100)
```

```
[1] 101
```

> **N.B.** Input arguments can be either **required** or **optional**. The latter have a fall-back default that is specified in the function code with an equals sign.
>
> Will this work? `add(100, 200, 300)`

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

```
Error in add(100, 200, 300): unused argument (300)
```

No!

This can be fixed with an ellipses (`...`) in the arguments when writing a function.

A second function

All functions in R look like this

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

The `sample()` function in R samples `size = n` values randomly from a vector `x`

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

```
[1] 5 8 7 2
```

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

```
sample(1:10, 12, replace = T)
```

```
 [1] 6 6 3 1 2 3 2 3 1 8 9 8
```

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

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

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

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

Version 1

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

> Q. Modify your function to return a FASTA-like sequence, so rather than [1] "G" "C" "A" "A" "T", we want "GCAAT"

Version 3 (FASTA format only)

```r
generate_dna_fasta <- function(n) {
  bases <- c("A", "T", "C", "G")
  dna <- sample(bases, size = n, replace = T)
  paste(dna, collapse = "")
}
```

> Q. Give the user an option to return FASTA format output sequence or standard multi-element vector format

Version 4 (FASTA format option)

```r
generate_dna <- function(n, fasta = T) {
  bases <- c("A", "T", "C", "G")
  dna <- sample(bases, size = n, replace = T)
  if (fasta) {
    dna <- paste(dna, collapse = "")
  }
  return(dna)
}
```

```r
generate_dna(35)
```

```
[1] "CTACTTGTCCAAAGACGACTTAGTCTGCTAAGCGT"
```

```r
generate_dna(35, fasta = F)
```

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

A new cool function

Q. Write a function called `generate_protein()` that generates a user specified length protein sequence in FASTA-like format?

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

```
generate_protein(70)
```

```
[1] "MNWVSVLVANFSYPPYVPYSPYHQYPCYAIYAEGYLLWDFTREGDFWLYYCTHDPMRTNYGGAPCVHENY"
```

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

```
library(rentrez)
seqs <- sapply(6:12, generate_protein)
```

```
for (i in 6:12) {
  cat(">", i, sep = "", "\n")
  cat(seqs[i - 5], "\n")
}
```

```
>6
PLETAR
>7
QTMHFYW
>8
```

```
CHVHPFHI
>9
CDYGARSVE
>10
YRCEALCMDY
>11
RIMLGCSAFDM
>12
EPYIRSCPDTGW
```

Copy and paste this into BLASTp