Class 6: R Functions

Joshua Khalil PID: A17784122

Table of contents

Background

All functions in R have at least three things:

- A **name** that we use to call a function.
- One or more input **arguments**.
- The **body** the line of R code that does the work

Our First Function

Let's write a little function called `add` to add some numbers (input argument)

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

}
```

Now we can use this function

```
add(x = 100, y = 1)
```

```
[1] 101
```

```
add(x = 10, y =10 )
```

```
[1] 20
```

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

```
[1] 101   2 101
```

Q. What if I give a multiple element vector to `x` and `y`?

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

```
[1] 200   2
```

Q. What if I give three inputs?

```
#add(x=c(100,1), y=1, z=1)
```

An error would appear as z is not defined as an input

Q. What if I give only one input to my function

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

}
```

```
addnew(x=100)
```

```
[1] 101
```

```
addnew(c(100,1), 100)
```

```
[1] 200 101
```

If we write our function with input arguments having no default value then the user will be required to set them when they use the function. We can give our input argument default values by setting them equal to some sensible value e.g. y=1…

A Second Function

Let's try something more interesting: Make a sequence generating tool...

The `sample()` function can be a useful start point

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

```
[1] 4 2 7 6
```

Q. Generate nine random numbers taken from input vector x=1:10?

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

```
[1]  4  8  2  3  1  5  6  7 10
```

Q. Generate twelve random numbers taken from input vector x=1:10?

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

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

Q. Write code for a `sample()` function that generates nuceleotide sequences of length 6

```
sample(x=c("A","T","G","C"), size =6, replace=TRUE)
```

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

Q. Write a first function `generated_dna()` that returns a user specified length DNA seqence

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

```
generated_dna(100)
```

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

Key-Points Every function in R looks fundamentally the same in term of its structure. Basically three components: Name, input, and body

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

Functions can have multiple inputs. These can be **required** arguments or **optional** arguments. With optional arguments having a set default value

Q Modify and improve our `generate_dna()` function to return its generated sequence in a more standard form "AGTAGTA" rather than the vector "A" "C" "G" "T"

```
generated_dna <- function(n = 6, fasta = TRUE) {

  ans <- sample(x = c("A","T","G","C"),
                size = n, replace = TRUE)

  if (fasta) {
    cat("Single-string output\n")
    ans <- paste0(ans, collapse = "")
  } else {
    cat("Multi-element vector output\n")
  }

  return(ans)
}


generated_dna()
```

```
Single-string output

[1] "TGGGAG"
```

The `paste()` function - it's job is to join up or stick input strings together

```
paste(c("alice","barry"), "loves R")
```

```
[1] "alice loves R" "barry loves R"
```

Flow control means where the R brain goes in your code

```
good_mood <- FALSE

if(good_mood){
  cat("Great")
} else{
  cat("Bummer")
}
```

```
Bummer
```

A protein Generating function

Q. Write a function that generates a user specifed length protein sequence

```
generate_protein <- function(n = 6, fasta = TRUE) {
  aa <- c("A","C","D","E","F","G","H","I","K","L",
          "M","N","P","Q","R","S","T","V","W","Y")  # 20 standard amino acids

  ans <- sample(aa, size = n, replace = TRUE)

  if (fasta) {
    ans <- paste0(ans, collapse = "")
  }

   return(ans)
}
generate_protein(42)
```

```
[1] "DVIMTNDQDYKSWRMQWCGQRMWWSWIFGFWGPTQRVTKGTQ"
```

Q. Use that function to generate random protein sequences between length 6 and 12

```r
for(i in 6:12){
  # FASTA ID line ">id"
  cat(">",i, sep="", "\n")
  # Protein sequence line
 cat(generate_protein(i), "\n")
}
```

```
>6
TYTDAC
>7
DWYIHRR
>8
ACGKCWVQ
>9
RMTYCLYHH
>10
FLVIQDGLLH
>11
EAPDTMWVSNY
>12
DKLISLNWDCTW
```

Q. Are any of the sequences unique or not found anywhere in nature?

Yes sequences 8, 9, 10, 11, and 12 are all unique.