Class 6: R functions

Jacob Hizon PID: A17776679

Table of contents

Background

All functions in R have at least 3 things:

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

Our first function

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

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

Now we can use this function

```
add(100, 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
```

What if I give three inputs to the function

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

Q. What if I give only one input to the add fucntion?

```
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 arguments "default" values by setting them equal to 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 starting point here:

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

```
[1] 10  8  2  4
```

Q. Generate 9 random numbers taken from the input vector x=1:10?

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

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

Q. Generate 12 random numbers taken from the input vector x=1:10?

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

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

Q. Write code for the `sample()` function that generates nucleotide sequence of length 6?

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

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

Q. Write a first fucntion `generate_dna()` that returns a *user specified length* DNA sequence

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

}
```

```
generate_dna(3)
```

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

```
generate_dna(
)
```

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

```
generate_dna(x=100)
```

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

Key-Points

Every function in R looks fundamentally the same in terms of its structure. Basically 3 things: 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_data()` function to return it's generated sequence in a more standard format "AGTAGTA" rahter than the vector "A", "C", "G", "A"

```
generate_dna <- function(x = 6, fasta = TRUE) {
  ans <- sample(c("A", "C", "T", "G"),
                                  size = x,
                                 replace = TRUE)
  if(fasta) {
    cat("Single-element vector output")
  ans <- paste(ans, collapse="") }
  else {
    cat("Multi-element vector output")}
  return(ans)
```

```
}

generate_dna(fasta = FALSE)
```

```
Multi-element vector output

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

The `paste()` function - it's job is to join up or stick together (a.k.a paste)

```
paste(c("alice", "loves R"), sep = "")
```

```
[1] "alice"   "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, called `generate_protein()` that generates a user-specified length protein sequecne.
>
> Q. Use that function to generate random protein sequences length 6 and 12
>
> Q. Are any of your sequences unique i.e. not found anywhere in nature
>
> Q. Write a function, called `generate_protein()` that generates a user-specified length protein sequecne. #The amino acids to sample from sample <-c("A", "R", "N", "D", "C", "Q", "G", "E", "H", "I", "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V") # Generate the function and specify size and replace size = x, replace = TRUE) # Remove the " " paste(ans, collapse="") }

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

```
generate_protein <- function(x) {
  ans <- sample(c("A", "R", "N", "D", "C", "Q", "G", "E", "H", "I", "L", "K", "M", "F", "P",
                                  size = x,
                                 replace = TRUE)
paste(ans, collapse="") }


generate_protein(12)
```

```
[1] "MDYVQDGGKFKM"
```

```
generate_protein <- function(x = 6) {
  ans <- sample(c("A", "R", "N", "D", "C", "Q", "G", "E", "H", "I", "L", "K", "M", "F", "P",
                                  size = x,
                                 replace = TRUE)
paste(ans, collapse="") }


generate_protein()
```

```
[1] "YHTACQ"
```

```
generate_protein <- function(x = 12) {
  ans <- sample(c("A", "R", "N", "D", "C", "Q", "G", "E", "H", "I", "L", "K", "M", "F", "P",
                                  size = x,
                                 replace = TRUE)
paste(ans, collapse="") }


generate_protein
```

```
function (x = 12)
{
    ans <- sample(c("A", "R", "N", "D", "C", "Q", "G", "E", "H",
        "I", "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V"),
        size = x, replace = TRUE)
    paste(ans, collapse = "")
}
```

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

```
>6
NLFNAE
>7
PRTPAEE
>8
MPVDPMIQ
>9
TFTLDLKMH
>10
EAPCHNWAIN
>11
GAQEALYYDMV
>12
LFCKRTAKDKVY
```

Q. Are any of your sequences unique i.e. not found anywhere in nature

Yes - there is uniqueness starting at 8aa for my generated protein sequences.