Class 6: R Function

Harshita Jha (PID:A17350910)

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 wee function called `add()` to add some numbers (the input arguments)

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

We have to run the code chunk before we can call the function

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

```
[1] 101
```

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

```
[1] 20
```

```
add(c(100,1,100),1)
```

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

```
add(100,c(100,1,100))
```

```
[1] 200 101 200
```

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 to the function?

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

R will show error because my function was written with only two inputs x and y, not z.

Q. What if I give only one input to the add function (only x or only y)?

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

In this addnew function, we have specified the default value of y.

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

```
[1] 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 arguments "default" values by setting them equal to some sensible value eg. y=1 in `addnew()` function.

A second function

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

The `sample()` functions can be a useful starting point here:

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

```
[1] 8 4 9 2
```

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

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

```
[1]  9 10  3  5  1  6  8  7  4
```

> Q. Can you generate 12 random numbers taken from input vector x=1:10?

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

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

For this question, we would have to change the replace input value to "TRUE" so that the code can run successfully.

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

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

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

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

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

```
generate_dna(length=25)
```

```
 [1] "A" "T" "C" "T" "A" "T" "T" "C" "A" "C" "T" "C" "T" "G" "A" "A" "T" "A" "C"
[20] "A" "A" "A" "G" "A" "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. Optional arguments have a default value set which is used automatically (user does not to specify it if they do not want to change the default).

Q. Modify and improve our `generate_dna()` function to return it's generated sequence in a more standard format like "AGTAGTA" rather than vector "A", "G","C","T"

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

if(fasta) {
  cat("Single element vector output")
ans <- paste(ans, collapse = "")
} else {
  cat("Multiple-element vector output")
}


return(ans)
}
```

```
generate_dna()
```

```
Single element vector output
```

```
[1] "AATGCG"
```

The `paste()` function - its job is to join up or stick together (a.k.a paste) input strings together.

```
paste("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 sequence.

```
generate_protein <- function(len=5) {

  # The amino acids to sample from
  aa <- c("A","R","N","D","C","Q","E","G","H","I","L","K","M","F","P","S","T","W","Y","V")
  # Draw n=len amino acids to make our sequence
  ans <- sample(aa, size= len, replace= TRUE)
  ans <- paste(ans, collapse="")
  return(ans)

  }
```

```
myseq <- generate_protein(len=42)
myseq
```

```
[1] "RYMMHLFRSYMDWVTQQLEITRETHGHTSNEADTFPLNTYYW"
```

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

```
generate_protein(6)
```

```
[1] "DEHLCR"
```

```
generate_protein(7)
```

```
[1] "GMNYFCW"
```

```
generate_protein(8)
```

```
[1] "DTSSTATW"
```

```
generate_protein(9)
```

```
[1] "RLNWPWDGH"
```

```
generate_protein(10)
```

```
[1] "MDKHPDYAAK"
```

```
generate_protein(11)
```

```
[1] "GTRCLCPWCGP"
```

```
generate_protein(12)
```

```
[1] "MHVNKNRIQKYH"
```

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

```
>6
FMMRDT
>7
PIHPKKR
>8
HWWFIQYP
>9
HCMIKIFNA
>10
QSHLKQPGGI
>11
CTDSHETGDGP
>12
VSLSFIYSNWLK
```

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

Yes, the sequences from 8 amino acids to 12 amino acids were unique. Sequence with 6 amino acids and 7 amino acids are not unique i.e they are already found in nature.