class06: R Function

Ethan Arevalo (A17393064)

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

Lets write a silly wee function called `add()` 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(c(100, 1, 100), 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 to the function

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

A Second Function

Lets 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] 2 5 8 4
```

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

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

```
[1]  1  9  3 10  7  5  8  4  2
```

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

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

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

Q. What if I give only one input to the odd 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 the 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 the equal to some sensible value

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

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

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

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

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

```
generate_DNA(len=10)
```

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

> **Key Points** Every function in R looks fundamentally the same in terms of its structure

```
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 format like "AGTAGT" rather than the vector "A", "G", "T", "C"

```
generate_DNA <- function(len=6, fasta=TRUE) {

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

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

  return(ans)
}

generate_DNA()
```

```
[1] "CGGGTA"
```

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

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

```
[1] "aliceloves R" "barryloves R"
```

Flow control means where the R brain goes in your code

```
good_mood <- FALSE

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

```
Bummer!
```

```r
generate_DNA <- function(len=6, fasta=TRUE) {

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

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

  return(ans)
}

generate_DNA()
```

```
Single-element vector output
```

```
[1] "TATCGC"
```

```r
generate_DNA(fasta=TRUE)
```

```
Single-element vector output
```

```
[1] "AAGTGA"
```

A Protein generating function

Q. Write a function, called `generate_protein()` that generates a user specified length protein sequence.

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

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

```r
generate_protein <- function(len) {

  # The amino acids to sample from
  aa <- c("A", "G", "T", "C", "R", "N", "D", "Q", "E", "H", "I", "L", "K", "M", "F", "P", "S"

  # Draw n=len amino acids to make our sequence
  ans <- sample(aa, size=len, replace=T)
  ans <- paste(ans, collapse = "")
  return(ans)
}
```

```r
aa <- c("A", "G", "T", "C", "R", "N", "D", "Q", "E", "H", "I", "L", "K", "M", "F", "P", "S",
```

```r
generate_protein(23)
```

```
[1] "FNFLSVITGGSKAPDYTVLGLVN"
```

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

```
>6
GMAVRF
>7
DSGADLI
>8
QGLLYEFH
>9
KPGQDPMPC
>10
APILQMHQMC
>11
NAVRPASKLSY
>12
NRNTGFLFWFWT
```