Class 6: R functions

Gavin Ambrose PID: A18548522

Table of contents

Background

All functions in R have at least 3 things:

- A **name** that we can 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
}
```

Now we can use this function:

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

```
[1] 101
```

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

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

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

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

```
[1] 200   2
```

> Q. What if I give three inputs to the function?

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

It doesn't work because we don't have a third variable in our function

> Q. What happens if I only give one input to the add function?

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

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

```
[1] 101
```

```r
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` in the `addnew()` function.

A second function

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

The `sample` function can be a useful starting point here:

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

```
[1]  2  1 10  5
```

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

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

```
[1]  5  4  3  2  9  1  6 10  8
```

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

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

```
 [1]  1  1  6  1  2  5  9  9  3  9  3 10
```

Yes, we must replace the "default" argument `replace = FALSE`.

> Q. Write code for the `sample` function thaat generates nucleotide sequence length of 6?

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

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

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

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

generate_dna(6)
```

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

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

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

Functions can have multiple inputs. These can be the **required** arguments or **optional** arguments, with optional arguments taking a default value.

> Q. Modify and improve our `generate_dna` function to return it's generated sequence in a more standard format such as "AGTA" instead of "A" "G" "T" "A"

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

generate_dna(6)
```

```
[1] "GCTAAT"
```

By using `paste()` and setting `collapse = "`, we remove the quotation marks in between each character entry.

TO remove the quotation marks in between each nucleotide, we would need to use `paste(x, collapse = "")` where x is the thing we want to paste.

Flow control means where the R brain goes in the control.

A protein generating function

> Q. Write a function that generates a user specified length protein sequence.

Generate an amino acid generator

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

Generate specific length sequences

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

Run the sequences through BLAST and check for 100% identity

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

generate_protein(6)
```

```
[1] "NMCVCA"
```

```r
generate_protein(7)
```

```
[1] "VMWWSPN"
```

```r
generate_protein(8)
```

```
[1] "VEHVLTDT"
```

```r
generate_protein(9)
```

```
[1] "EYTMFQVYL"
```

```r
generate_protein(10)
```

```
[1] "KEAIQLKSTM"
```

```r
generate_protein(11)
```

```
[1] "WYNFWQDNLHY"
```

```r
generate_protein(12)
```

```
[1] "DSIWEKDSSLRN"
```

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

```
>6
ETHWSF
>7
IVMLGTR
>8
WWRDSQPS
>9
YCHMYNCWM
>10
PWVAGMNSDC
>11
IFDTIQAQMCQ
>12
PCIFDAIAGGKG
```

I got 100% identity for proteins of length 6, 7, and 8. There is not 100% identity for proteins of length 9, 10, or 11.

Takeaway:

Keep your spirits up while creating a function. Aim for a functional but limited code, and add the bells and whistles to the already functional code.