Class 6: R functions

Melanda Aboueid (A17473102)

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 argumentd)

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

Now we can use this function

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

```
[1] 101
```

```
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 to the function?

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

Q. What happens if I give only one input to the add 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 than the user will be required to set them when the use the function> We can give our input arguments "default" valuse

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]  5  7 10  1
```

Q. Generating 9 random numbers taken from the input vectore x=1:10?

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

```
[1] 2 5 1 4 9 8 3 6 7
```

Q. Generating 12 random numbers taken from the input vectore x=1:10?

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

```
 [1]  1  2 10  2  9  7  7 10  1  2  4  1
```

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

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

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

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

```r
generate_dna <- function(len=100) {
  sample(c("A", "C", "G", "T"), size =len, replace= TRUE)
}
```

```r
generate_dna(len=100)
```

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

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_dna` function to return it's generated sequence in a more standard format like "AGTAGTA" rather than vector "A", "T", "C","G"

```
generate_dna <- function(len=6, fasta=TRUE) {
  ans <- sample(c("A", "C", "G", "T"),
               size =len, replace= TRUE)
  if(fasta) {
    ans <- paste(ans, collapse= "")
  }
 return(ans)
}
generate_dna()
```

```
[1] "CTACTC"
```

The `past()` function - it's job ia 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 <- TRUE

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

```
GREAT!
```

A protien generating function

> Q. Write a function that generates user specified lengh prtien sequence.

> Q. Use that function to generate protien sequences between length 6 and 12

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

There are 20 natural amino acids

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

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

   # 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= T)
  ans <- paste(ans, collapse="")
  return(ans)
}
```

```r
myseq <-generate_protien(142)
myseq
```

```
[1] "WFRNCHLRHPINHLPNIGDRTMRDWMACWCEDLRHYVGVICLEIQCAPPPQGSEFEPPILWYHFQHQKPEIWWVSFDMKTRYNPHGG
```

> Q. Use that function to generate protien sequences between length 6 and 12

```r
generate_protien(6)
```

```
[1] "TSFPCF"
```

```r
generate_protien(7)
```

```
[1] "LQMIMDS"
```

```r
generate_protien(8)
```

```
[1] "RYDVCMYL"
```

```r
generate_protien(9)
```

```
[1] "FLGLVVMKQ"
```

```r
generate_protien(10)
```

```
[1] "LWFNHHYDTV"
```

```r
generate_protien(11)
```

```
[1] "GKSVNFPKVEY"
```

```r
generate_protien(12)
```

```
[1] "QRKVELINDTMS"
```

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

```
>6
FFTGGS
>7
DDCSEEW
>8
TALHFMVH
>9
YQICFLMKM
>10
VFAYYHERYE
>11
MWWSCAMELWL
>12
NMMMVNSMIQEM
```