

class 06 R functions

Ryan Bench (PID:A69038034)

Table of contents

Our first (silly) function	1
A second function	2
A protein generating function	3

All functions in R have at least 3 things

- A **name**, we pick this and use it to call our function,
- Input **arguments** (there can be multiple)
- The **body** lines of R code that do the work

Our first (silly) function

Write a function to house some numbers

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

If we have (x, y=1), if y is not defined it just defaults to 1 Now we can call the function:

```
add(c(10, 10), 100)
```

```
[1] 110 110
```

A second function

We are writing a function that will generate random nucleotide sequences of a user specified length

The `sample` function can be helpful here

```
sample(c("A", "C", "T", "G"), size=50, replace = TRUE)
```

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

I want a 1 element long character vector that looks like this “CACAGC” not “C” “A” “C” “A” “G” “C”

```
v <- sample(c("A", "C", "T", "G"), size=50, replace = TRUE)
paste(v, collapse = "")
```

```
[1] "AAACCTCGCCTGGTTTCATTCGGCGTTTATACACTCGCCGGTGCCTAGACC"
```

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

Test it:

```
generate_DNA(100)
```

```
[1] "ATCACTATATACACGTAGGCTGAGGACTCTGGAGTGGCGGAGTCTACCGTCCCTACATTCTGGAGCTCCGTGTCTACTTACGGTGAGT"
```

```
fasta <- FALSE
if(fasta) {
  cat("HELLO You!")
} else {
  cat("No you dont!")
}
```

No you dont!

Add the ability to return a multi-element vector or a single element fasta like vector.

```
generate_fasta <- function(size = 50, fasta = TRUE) {  
  v <- sample(c("A", "C", "T", "G"), size = size, replace = TRUE)  
  s <- paste(v, collapse = "")  
  
  if(fasta) {  
    return(s)  
  } else {  
    return(v)  
  }  
}
```

```
generate_fasta(60, fasta = TRUE)
```

```
[1] "GCACTGCTCAATCTGCCCCGAGTTAATATTGTGTCTCAGCCTAAGCCCCGCAGGCTAGTC"
```

```
generate_fasta(60, fasta = FALSE)
```

```
[1] "A" "A" "C" "G" "A" "C" "T" "T" "T" "G" "C" "C" "C" "C" "T" "G" "A" "T" "T"  
[20] "C" "A" "A" "A" "G" "T" "A" "T" "G" "A" "A" "A" "T" "C" "A" "T" "C" "A" "G"  
[39] "C" "A" "A" "C" "A" "G" "A" "T" "T" "T" "T" "G" "G" "C" "A" "A" "A" "C" "G"  
[58] "A" "A" "G"
```

A protein generating function

```
generate_protein <- function(size = 50, fasta = TRUE) {  
  aa <- c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I", "L", "K", "M", "F", "P", "S", "T")  
  v <- sample(aa, size = size, replace = TRUE)  
  s <- paste(v, collapse = "")  
  
  if(fasta) {  
    return(s)  
  } else {  
    return(v)  
  }  
}
```

```
generate_protein(60)
```

```
[1] "LTWFMCTTACIFATTTGLYWFANDWNHWIRSNQRTEHPNNKCDIGYRMSQGDFSGTVYLH"
```

Use our new “generate_protein()” function to make random protein sequences of length 6-12 (i.e. one length 6, one length 7, etc up to length 12)

This can be done via “brute force”

```
generate_protein(6)
```

```
[1] "NMEIHA"
```

```
generate_protein(7)
```

```
[1] "DLKAEES"
```

```
generate_protein(8)
```

```
[1] "EQHSNDMM"
```

```
generate_protein(9)
```

```
[1] "MTNPTNFHN"
```

```
generate_protein(10)
```

```
[1] "QQGASPFNMW"
```

```
generate_protein(11)
```

```
[1] "DVFWMRFWMPD"
```

```
generate_protein(12)
```

```
[1] "GLKSVTNFVYVW"
```

A second way to do this is to use the `for()` loop:

```
lengths <- 6:12
lengths
```

```
[1] 6 7 8 9 10 11 12
```

```
for(i in lengths) {
  cat(">", i, "\n", sep="")
  aa <- generate_protein(i)
  cat(aa)
  cat("\n")
}
```

```
>6
KIFCPE
>7
CFSFCFS
>8
YVRPMDMK
>9
IWMWTWQRE
>10
GRGCTCINKM
>11
QVPWLMLTKFM
>12
IWGMTQWADHRR
```

```
sapply(6:12, generate_protein)
```

```
[1] "RCDSES"      "QDKYIQQ"      "PFIQFRQP"      "RRMVGYAGD"      "PTKMYQHQRVP"
[6] "QIMRWSRGHVH" "ESNHGLVHAMCC"
```