

Class 06: R Functions

Laura Sun (PID: A17923552)

All functions in R have at least 3 things:

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

Our first wee function:

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

Let's test our function

```
add(c(1,2,3), y=10)
```

```
[1] 11 12 13
```

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

```
[1] 110
```

A second function

Let's try something more interesting. Make a sequence generation tool.

The `sample()` function could be useful here.

```
sample(1:10, size=3)
```

```
[1] 1 9 3
```

Change this to work with the nucleotides ATCG and return 3 of them.

```
sample(c("A", "T", "C", "G"), size=3)
```

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

Or I can name them as n and sample(n, size=3).

With the above code, I can't generate more than 4 (sample larger than the population). Try this instead, replace (sampling be with replacement). Default is FALSE, we set to TRUE.

```
n <- c("A", "T", "C", "G")
sample(n, size=10, replace = TRUE)
```

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

Turn this snippet into a function that returns a user specified length DNA sequence. Let's call it `generate_DNA`.

```
generate_DNA <- function(len=10){
  n <- c("A", "T", "C", "G")
  sample(n, size=len, replace = TRUE)
}
```

or

```
generate_DNA <- function(len=10, fasta=FALSE){
  n <- c("A", "T", "C", "G")
  v <- sample(n, size=len, replace = TRUE)
  # make a single element vector
  s <- paste(v, collapse="")
  cat("Well done!\n")
  if(fasta){
    return(s)
  }
  else{
    return(v)
  }
}
```

```
s <- generate_DNA(15)
```

Well done!

```
s
```

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

I want the option to return a single element character vector with my sequence all together like this: “GGAGTAGC”.

```
s
```

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

```
paste(s, collapse="")
```

```
[1] "TGTGCCACGACCCAC"
```

```
generate_DNA(10, fasta=TRUE)
```

Well done!

```
[1] "GTCGAATCCA"
```

A more advanced example

Make a third function that generates protein sequence of a user specified length and format.

```
proseq <- function(len=10, fasta=FALSE){  
  aa <- c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I",  
         "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V")  
  seq <- sample(aa, size = len, replace = TRUE)  
  if (fasta) {  
    return(paste(seq, collapse = ""))  
  }  
  else {  
    return(seq)  
  }  
}
```

```
proseq(15, TRUE)
```

```
[1] "MNDELMGHRSCHYHFA"
```

Generate random protein sequences between lengths 5 and 12 amino acids.

Our approach is to do this by brute force calling our function for each length 5 to 12.

Another approach is to write a `for()` loop to iterate over the input valued 5 to 12.

A very useful third R specific approach is to use the `sapply()` function.

```
seq_length <- 6:12
for (i in seq_length) {
  cat(">", i, "\n")
  cat(proseq(i))
  cat("\n")
}
```

```
> 6
I G T A W K
> 7
H M I K P H F
> 8
Y F Y I K W R H
> 9
H Y H C I E K N V
> 10
S D L S L P D R Y F
> 11
V M T C H V S E R M M
> 12
V D T R D E D T S V N R
```

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

```
[[1]]
[1] "C" "S" "E" "D" "N" "N"

[[2]]
[1] "L" "S" "P" "I" "G" "P" "E"
```

```
[[3]]  
[1] "M" "H" "R" "G" "N" "S" "G" "E"  
  
[[4]]  
[1] "L" "V" "I" "Y" "G" "Y" "R" "W" "P"  
  
[[5]]  
[1] "D" "E" "D" "L" "P" "E" "A" "I" "K" "V"  
  
[[6]]  
[1] "R" "N" "E" "N" "S" "W" "Q" "R" "R" "N" "C"  
  
[[7]]  
[1] "E" "M" "K" "V" "W" "A" "A" "H" "Q" "H" "V" "N"
```

Key-point: writing functions in R is doable but not the easiest thing in the world.
Starting with a working snippet of code and then using LLM tools to improve and
generalize your function code is a productive approach.