Class 6: R Functions

Shazreh Hassan (PID: A13743949)

Table of contents

All functions in R have at least 3 things:

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

First silly function

Write a function to add some numbers:

```
#in this case, the default for y will be 1 for when user doesn't specify a value for y
add <- function(x, y=1) {
  x+y
}
```

Now we can call this function:

```
add(10,5)
```

```
[1] 15
```

```
#can add to a vector
add(c(10,10),100)
```

```
[1] 110 110
```

A second function

Write a a function to generate random nucleotide sequences of a user specified length:

The `sample()` function can be helpful here; it samples randomly from a vector.

```
sample(c("A","C","G","T"), size=100, replace=TRUE)
```

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

I want a 1 element long character vector that looks like FASTA format: "TATTTA" instead of having quotes and spaces in between

```
#default for collapse is to have a space
v <- sample(c("A","C","G","T"), size=50, replace=TRUE)
paste(v, collapse = "")
```

```
[1] "ATGCCCGCCTCACCAGTTGTAACCACCCTCTGTTTACGTCAAAAAACCTA"
```

Turn this into a function:

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

Test it:

```
generate_dna(60)
```

```
[1] "ACGCACCGACCTTACTCAAGTAACGGCACCTTACGTATACTACATAAGTGAATGTCAAGG"
```

if else Functions

```
if(TRUE) {
  cat("HELLO You!")
}
```

```
HELLO You!
```

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","T","C","G"),size=size, replace = TRUE)
  s <- (paste(v, collapse=""))

if(fasta) {
  return(s)
} else {
  return(v)
}
}

generate_fasta(8)
```

```
[1] "CTGCGATG"
```

Now change from DNA sequence to protein sequence:

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

if(fasta) {
  return(s)
} else {
  return(v)
}
}

generate_protein(9)
```

```
[1] "YLVESHRDP"
```

Use the new `generate_protein()` function to make random protein sequences of length 6 to 12 (i.e. one length 6, one length 7, etc. up to 12).

Use a `for()` loop

```
# \n is for new line
lengths <- 6:12

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

```
>6
DYLGWA
>7
IHLAMED
>8
VVMDMNSK
>9
MASDTRTSI
>10
SSAFYYQFAT
>11
DRFRHGAVEYT
>12
DQSSDWHFNALI
```

A better way to solve this is to use the `apply()` family of functions, specifically the `sapply` function in this case.

```
sapply(lengths, generate_protein)
```

```
[1] "AQEAYQ"       "QCSYEKY"      "IWTCVDKV"     "KFPALQTFG"    "GDCNYHLRCK"
[6] "ISGNDKTLNVD"  "NSGMCQFMNSLK"
```