Class 6: R functions

Jennifer Thai (PID: A17893762)

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)
```

```
[1] 11
```

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

```
[1] 20
```

A second function

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

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

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

```
[1] 10  8  4
```

Change this to work with the nucleotides A, C, G, and T, and return 3 of them

```
n <- c("A", "C", "G", "T")
sample(n, size=15, replace=TRUE)
```

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

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, fasta=FALSE) {
  n <- c("A", "C", "G", "T")
  v <- sample(n, size=len, replace=TRUE)

  # Make a single element vector
  s <- paste(v, collapse="")

  cat("Well done you!\n")

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

```
generate_dna(5)
```

```
Well done you!
```

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

```
s <- generate_dna(15)
```

```
Well done you!
```

```
s
```

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

I want the option to return a single element character vector with my sequence all together like this: "GGAGTAC"

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

```
Well done you!
```

```
[1] "TCCAGTAAAA"
```

```
generate_dna(10, fasta=FALSE)
```

```
Well done you!
```

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

A more advanced example

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

```
a <- c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I", "L", "K", "M", "F", "P", "S", "T",
sample(a, size=15, replace=TRUE)
```

```
 [1] "C" "A" "P" "S" "F" "V" "S" "V" "A" "L" "E" "H" "I" "N" "W"
```

```
generate_protein <- function(len=10, fasta=TRUE) {
  a <- c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I", "L", "K", "M", "F", "P", "S", "T"
  seq <- sample(a, size=len, replace=TRUE)

  p <- paste(seq, collapse="")

  if(fasta) {
    return( p )
  } else {
    return ( seq )
  }
}
```

```
generate_protein(10)
```

```
[1] "WLYRVNSYHI"
```

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

```
generate_protein(5)
```

```
[1] "THNPG"
```

```
generate_protein(6)
```

```
[1] "VSQPQH"
```

One 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_lengths <- 5:12
for (i in seq_lengths) {
  cat(">", i, "\n", sep="")
  cat( generate_protein(i) )
  cat("\n")
}
```

```
>5
KHDRF
>6
KEILMS
>7
LKWMVAE
>8
RDYARTKH
>9
CNVVYANES
>10
RPDAPEGVCL
>11
GINRNAFQPVR
>12
FLILEFPWGDPT
```

```
sapply(5:12, generate_protein)
```

```
[1] "EHRPV"        "VLVTLK"       "YVRCCVA"      "QDHGWLCT"     "DGWDYAMKM"
[6] "CYCRPDWIEP"   "RQEQMIWKHIG"  "DLTGFFPVTRMP"
```

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