Class 6: R Functions

Peyton Ciu (PID:A18145937)

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 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

```
n <- c("A","C","G","T")
sample(n,replace =TRUE,  5)
```

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

Turn this snippet into a function that returns a user specified length dna sequence. Let's call it 'generate_dna'

```
generate_dna <-function(x, fasta =FALSE) {
  n <- c("A","C","G","T")
  # Makes single element polynucleotide sequence and stores in a vector
 v<- sample(n,replace =TRUE,  x)

 # Makes gets rid of quotations
  s<- paste(v, collapse="")

  cat("Well done you! \n")

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

```
generate_dna(5)
```

```
Well done you!
```

```
[1] "CAGAA"
```

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

```
Well done you!
```

```
s
```

```
[1] "TCTCACAAACCACAA"
```

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

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

```
Well done you!
```

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

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

```
Well done you!
```

```
[1] "CGGGTGATGC"
```

A more advanced example

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

```
generate_protein <- function(x, vector =FALSE){
amino_acids <- c("A", "R", "N", "D", "C",
                 "Q", "E", "G", "H", "I",
                 "L", "K", "M", "F", "P",
                 "S", "T", "W", "Y", "V")
vector_form<-sample(amino_acids, x, replace=TRUE)
protein_seq <-paste(vector_form, collapse = "")
if(vector ==TRUE){
    return(vector_form)
    }else {
  return(protein_seq)}
}
```

```
generate_protein(20)
```

```
[1] "GTYHPTAKLCQEEGYEGILE"
```

```
generate_protein(20, vector =TRUE)
```

```
 [1] "K" "V" "N" "P" "V" "N" "M" "P" "V" "Y" "G" "G" "G" "Q" "V" "D" "E" "V" "T"
[20] "L"
```

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

```
seq_lengths<- 6:12
for (i in seq_lengths) {
  cat(">", i, "\n", sep="")
  cat(generate_protein(i))
  cat("\n")
}
```

```
>6
MNTIVA
>7
TLNYRTA
>8
RKLHHSEF
>9
CLAHCRKFY
>10
HVVGWRMRNI
>11
FKLVQLIVVFH
>12
NNYGCMGQNGDA
```

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

Another is to write a 'for()' loop to iterate through 5 to 12

A very useful 3rd approach is the 'sapply()' function

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

```
[1] "YEAIS"        "ICMGAI"       "LFDYVAL"      "ICFLRWWL"     "NPEHWYAEY"
[6] "YVIIDQAMGF"   "THWFYCMDHLF"  "KGHQSWTCIHAP"
```

Keypoint writing functions in R is doable but not the easiest thing in the world. Start with a working snippet of code and then using LLM tools to improve and generalize your code is a good approach.