Class 06: Functions

Vanesa Fernandez (PID: A59026769)

The first function we write

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

Can we use it?

```
add(1,1)
```

```
[1] 2
```

```
add(x=1,y=100)
```

```
[1] 101
```

```
add(c(100,1,100),1)
```

```
[1] 101   2 101
```

If you wanna define a default value, such as y=1, then we can omit the y value on the function.

```
add2 <- function(x,y=1){ x +y
}
```

```
add2(10)
```

```
[1] 11
```

#A second function Q1. Make a function "generate_DNA()" that makes a random nucleotides seq of any lenght

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

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

The last one is out wee working snipet. Now let's try make it into a function.

```
generate_DNA <- function(length){
bases <- c("A","C","G","T")
sequence <- sample(bases, size = length,
         replace=TRUE)
  return(sequence)
}
```

```
generate_DNA(10)
```

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

After installing the package bio3d, we can access the table of aa

```
bio3d::aa.table$aa1
```

```
 [1] "A" "R" "N" "D" "C" "Q" "E" "G" "H" "I" "L" "K" "M" "F" "P" "S" "T" "W" "Y"
[20] "V" "X" "D" "R" "C" "C" "C" "C" "C" "C" "C" "C" "H" "E" "H" "H" "H" "H" "H"
[39] "H" "D" "K" "K" "M" "K" "M" "C" "F" "Y" "S" "T"
```

To make it unique:

```
 aa <- unique(bio3d::aa.table$aa1)[1:20]
aa
```

```
 [1] "A" "R" "N" "D" "C" "Q" "E" "G" "H" "I" "L" "K" "M" "F" "P" "S" "T" "W" "Y"
[20] "V"
```

Then we can write a function that generates proteins

```r
generate_protein <- function(length){ aa
  aachain <- sample(aa, size = length,
         replace=TRUE)
  aachain <- paste(aachain, collapse="") #we added the paste function in order
  return(aachain)
  }
```

```r
generate_protein(10)
```

```
[1] "TFKNRFMVLC"
```

Q. Generate random protein sequences of length 6 to 13.

```r
generate_protein(6)
```

```
[1] "SHEIHA"
```

```r
generate_protein(7)
```

```
[1] "SPAPWSV"
```

```r
generate_protein(8)
```

```
[1] "YIWCNFEY"
```

```r
generate_protein(9)
```

```
[1] "FHHYWSIRY"
```

```r
generate_protein(10)
```

```
[1] "SLGNDPQYSM"
```

```r
generate_protein(11)
```

```
[1] "NLCFRGHNMAV"
```

```r
generate_protein(12)
```

```
[1] "RIDVFTHSVILQ"
```

```r
generate_protein(13)
```

```
[1] "DPTFNLIFFPEPF"
```

Then to write it shorter

```r
 for (x in 6:13) print(generate_protein(x))
```

```
[1] "HEQALD"
[1] "YCVVILG"
[1] "IGEEPEAV"
[1] "RQLMYQGQN"
[1] "SSSFYPKVHL"
[1] "QQSNEFICCAF"
[1] "HENNTENMQCNW"
[1] "KFSHNSGFDEHFD"
```

In class strategy:

```r
 X <- c(6:13)
answer <- sapply(X, generate_protein)
```

To get the fasta format, we can just add the headed to each sequence that we generated.

```r
cat( paste(">id.",6:13, "\n", answer, sep=""), sep="\n")
```

```
>id.6
TIFWNW
>id.7
DDGVMSA
>id.8
YEIYPYLD
>id.9
QALNWGVVY
```

```
>id.10
PSSLMCCCTT
>id.11
RKRSYDLMQRE
>id.12
MQMRPGECLFWK
>id.13
TSRAELKCFALLN
```

Functions: Sintax

```r
name_of_function <- function (input1, input2)
{
  #Function Body
  output <- input1 + input2

  return (output) ## not necessarily needed it
}
```