Class 6: R Functions

Seona Patel (PID: A69035519)

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** (there can be multiple)
- The **body** lines of R code that do the work

Our first (silly) function

Write a function to add some numbers

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

Now we can call this function:

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

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

```
[1] 110
```

A second function

Write a function to generate random nucleotide sequences of a user specified length: The `sample()` function can be useful here.

```
sample(c('A','C','T','G'), size=30, replace=TRUE)
```

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

I want a 1 element long character vector that looks like "GACTA"

```
v <- sample(c('A','C','T','G'), size=30, replace=TRUE)
paste(v,collapse='')
```

```
[1] "TAAGACTAAAAGTTAAGGTCGCACCGTTCG"
```

```
generate_dna <- function(size=50) {
v <- sample(c('A','C','T','G'), size=size, replace=TRUE)
paste(v,collapse='')}
```

Test it:

```
generate_dna(60)
```

```
[1] "TAGAATGCAGGTAGGCCCGTTATCTTACAATACATTTCACAGTGACCAACTTGGTCCTCA"
```

```
fasta <- FALSE
if(fasta) {
  cat("HELLO You!")
} else {
    cat("No you dont")
  }
```

```
No you dont
```

Add the ability to return a multi-elemnt vector or a single element fasta like vector.

```r
generate_fasta <- function(size=50, fasta=TRUE) {
v <- sample(c('A','C','T','G'), size=size, replace=TRUE)
s <- paste(v, collapse='')
if(fasta) {
return(s)
}
else{
return(v)}
}
```

```r
generate_fasta(50, TRUE)
```

```
[1] "CGGATCCAAGACATCCGAAGGAGGTAGTGGCGAGGCTCATTTCACTAAGT"
```

```r
generate_fasta(50, FALSE)
```

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

A protein generating function

```r
generate_protein <- function(size=50, fasta=TRUE) {
  aa <- c('A','R','N','D','C','Q','E','G','H','I','L','K','M','F','P','S','T','W','Y','V')
  v <- sample(aa, size=size, replace=TRUE)
  s <- paste(v, collapse='')
  if(fasta) {
    return(s)
  } else {
    return(v)
  }
}
```

```r
generate_protein(6)
```

```
[1] "EWRFIC"
```

Use new `generate_protein()` function to generate random protein sequences of lengths between 6 and 12.

One way to do this is brute force.

A second way is usign a `for()` loop:

```r
lengths <- 6:12
lengths
```

```
[1]  6  7  8  9 10 11 12
```

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

```
>6
VSITGD
>7
SRMCQFS
>8
PNCIFHNN
>9
MCHMGTAMA
>10
FQYNPNSFTG
>11
IFWKEWVTGWF
>12
INMPCVDYQASH
```

```r
paste(c('barry', 'monika'), "R", sep=" loves ")
```

```
[1] "barry loves R"  "monika loves R"
```

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

```r
sapply(6:12, generate_protein)
```

```
[1] "WDRCNA"       "EAWFFHR"      "MWQIFFPL"     "PQLYACFIN"    "NFMAPYDAVN"  
[6] "WRRKATQQVPT"  "PQNSSRRWNVGQ"
```