Class 6: R Functions

Alexandra Garcia (PID: A16278166)

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 tod 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, 100)
```

```
[1] 110
```

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

```
[1] 110 110
```

< You can have arguments with a default value by setting the value with =

A second function

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

The `sample()` function can be useful here.

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

```
[1] "CCCGTGAACCAAACAATCTAGATGTCACTCGATGAGGTTTTGAGGACCCG"
```

Turned this into my first function:

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

Test it:

```
generate_dna(6)
```

```
[1] "TCAATG"
```

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

```
fasta = TRUE
if(fasta) {
  cat("Hello")
} else{
  cat("Bye")
}
```

```
Hello
```

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

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

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

```
[1] "TCTTTATACT"
```

A protein generating function

So you have to make amino acids

```
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
  v <- sample(aa, size=size, replace=TRUE)
  s <- paste(v, collapse ="")

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

```
generate_protein(6)
```

```
[1] "IRIDHD"
```

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

One way to do it is brute force

```
generate_protein(6)
```

```
[1] "RLQDEF"
```

```
generate_protein(7)
```

```
[1] "TSDMFRA"
```

A second way to do it is to use a `for()` loop:

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

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

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

```
>6
VCQAYWVCQAYW
>7
QLTTWEEQLTTWEE
>8
LGWVYCWHLGWVYCWH
>9
DNESDKWVPDNESDKWVP
>10
LTGRNKDDGRLTGRNKDDGR
>11
HKYLPIHSDDGHKYLPIHSDDG
>12
NKRDMNETPLNGNKRDMNETPLNG
```

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] "GYQMGA"       "RQPLMSA"      "VDEHVIED"     "SQKSVAPAT"    "CTFQCRQQMY"
[6] "WHPSGQDINAR"  "PMYHTPFGIDSS"
```