Class 6: R functions

Rachel Felix, PID: A16037641

Table of contents

All function 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(c(10,10),100)
```

```
[1] 110 110
```

```
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 helpful here.

```
sample(c("A", "C", "G", "T"), size = 100, replace= TRUE)
```

```
  [1] "C" "T" "G" "T" "G" "A" "G" "G" "T" "C" "A" "C" "A" "A" "G" "C" "G" "A"
 [19] "G" "T" "T" "G" "C" "G" "A" "A" "T" "T" "G" "C" "C" "A" "C" "A" "T" "C"
 [37] "C" "G" "A" "A" "T" "A" "A" "C" "A" "C" "T" "A" "G" "C" "T" "G" "G" "C"
 [55] "A" "A" "T" "T" "T" "G" "G" "G" "C" "G" "T" "T" "T" "G" "C" "G" "T" "T"
 [73] "C" "C" "A" "T" "T" "A" "C" "T" "G" "A" "A" "A" "A" "C" "C" "C" "G" "G"
 [91] "C" "A" "C" "C" "A" "A" "G" "T" "T" "C"
```

I want the a 1 element long character vector that looks “CACAGC” no “C” “A” “C” “A” “G” “C”

```
v <- sample(c("A", "C", "G", "T"), size = 100, replace= TRUE)
paste(v, collapse = "")
```

```
[1] "GCGTTACTTGTTAGCCTAATATGTTATCAAATTGCTTTTGTAGGAGCTAGTTTTCAGCAGCTAGGTTAGTACTACTTATGCTGCGTAC
```

Turn this into my first wee function

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

Test it:

```
generate_dna(60)
```

```
[1] "GTTTTCTGTCGAGACAGAGATTCAAATAATCAAATTATGCTTGGGTAGCCAGGAACGCTT"
```

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

```
HELLO You!
```

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

```
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)
```

```
[1] "TCGAAAAGAC"
```

```
generate_fasta(10,)
```

```
[1] "TCTTAGAATA"
```

A protein grading function

```
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)
  }
}
```

```
generate_protein(6)
```

```
[1] "LMYSTN"
```

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

One way to do this is "brute force"

```
generate_protein(6)
```

```
[1] "TQPTQN"
```

```
generate_protein(7)
```

```
[1] "WYAGCCP"
```

```
generate_protein(8)
```

```
[1] "INRGHSHA"
```

```
generate_protein(9)
```

```
[1] "HFRNCDYGV"
```

A second way 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 = "")
  aa <- generate_protein(i)
  cat(aa)
  cat("\n")
}
```

```
>6
YVYNCS
>7
EPYHAKD
>8
DYYYGCSE
>9
KGSYLNKMH
>10
NVSTQGWQRP
>11
KHYFLEIEKRS
>12
VSTKCLEANYCG
```

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

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

```
[1] "SPCWET"       "MSWEWFM"      "HDGTEKSY"     "VYTWHFQMM"    "QRIVEIMHMC"
[6] "SDFETVTMKFK"  "TGHRDCRHHHTG"
```