Class 6: R Function

Wade Ingersoll (PID: A69038080)

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(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=50, replace=TRUE)
```

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

I want a 1-element long character vector that looks like: "CCAGTC", not "C" "C" "A" "G" "T" "C"

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

```
[1] "ATGCCAAAATTGCGCAGCTCCTGACAATCAGAAATACGTATCGCACGAGG"
```

Turn 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(30)
```

```
[1] "TCCGAAATGCTCAATCCACTTTGATTCCTT"
```

Practicing IF:

```
fasta <- TRUE
if(fasta) {
  cat("HELLO You!")
} else {
  cat("No you don't!")
}
```

```
HELLO You!
```

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

```
generate_dna <- 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)
  }
}
```

Try it:

```
generate_dna(fasta = TRUE)
```

```
[1] "TCTACGCAGGGGTGGACATTAAACATAGGATTCATTGATATGCTGTGAGT"
```

```
generate_dna(fasta = FALSE)
```

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

Protein generating 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)
  }
}
```

Try it:

```
generate_protein(10)
```

```
[1] "DVMYGRKFNF"
```

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 length 12)

One way to do this is "brute force"

```
generate_protein(6)
```

```
[1] "THCSCH"
```

```
generate_protein(7)
```

```
[1] "LNALIVY"
```

```
generate_protein(8)
```

```
[1] "AGWEHYFD"
```

```
generate_protein(9)
```

```
[1] "NPTMTCASG"
```

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
RQAGGC
>7
ATPCPFP
>8
GSPVIRGM
>9
QEPNITYGI
>10
DPWVTMLRCP
>11
EPAAKMWSGIS
>12
AITIETEPVADH
```

Try `sapply()` function

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

```
[1] "DEYYTQ"       "DNTSIVE"      "RYEFGLWW"     "ERAGTFCLC"    "GHQKREWQCF"
[6] "GMMWDSVLCFQ"  "RWDIKKRDQSWW"
```