Class 6: R Functions

Kavi (PID: A69046927)

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, 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.

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

I want the a 1 element leong characyer vector that looks like "CACAGC", not "C","A","C","A","G","C".

```
paste(v,collapse="")
```

```
[1] "GACACTCGTTGAGATCAAAGTCAGAACAGCATAAAGCGATGTGAGGATAA"
```

Combined, the code is:

```
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] "CCTGTGCAGCCGTTCGTCTTGACCAGTACCAGCCGGTGTTAGTTGGTAGGCTAAGTCGAA"
```

```
fasta <- FALSE
if(TRUE) {
  cat("HELLO You!")
}
```

```
HELLO You!
```

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

```
fasta <- FALSE
generate_fasta <- function(size=50) {
  paste(sample(c("A","C","G","T"), size = size, replace = TRUE),collapse="")
}
generate_me <- function(size=50) {
  sample(c("A","C","G","T"), size = size, replace = TRUE)
}
if(fasta == TRUE) {
  generate_fasta()
} else{
  generate_me()
}
```

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

Now to generate a protein sequence...

```
fasta <- FALSE
generate_fasta <- function(size=50) {
  paste(sample(c("A","R","N","D","C","Q","E","G","H","I","L","K","M","F","P","S","T","W","Y"
}
generate_me <- function(size=50) {
  sample(c("A","R","N","D","C","Q","E","G","H","I","L","K","M","F","P","S","T","W","Y","V"),
}
if(fasta == TRUE) {
  generate_fasta()
} else {
  generate_me()
}
```

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

Better way:

```
generate_protein <- function(size = 50, fasta = FALSE) {
  aa <- c("A","R","N","D","C","Q","E","G","H","I","L","K","M","F","P","S","T","W","Y","V")
  seq <- sample(aa, size = size, replace = TRUE)
  if(fasta) {
    return(paste(seq, collapse = ""))
  } else {
    return(seq)
  }
}
```

```
generate_protein(60, fasta = FALSE)
```

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

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 a length of 12.)

One way to do this is "brute force".

```
generate_protein(6)
```

```
[1] "N" "K" "F" "N" "E" "E"
```

```
generate_protein(7)
```

```
[1] "Y" "L" "D" "L" "R" "R" "H"
```

```
generate_protein(8)
```

```
[1] "K" "M" "G" "W" "G" "H" "L" "L"
```

```
generate_protein(9)
```

```
[1] "N" "Y" "S" "A" "C" "P" "R" "C" "P"
```

Work smarter, not harder:

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

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

```
>6
CFWQTR
>7
HYKMSYL
>8
RPEGFEVK
>9
DSRVPRCPL
>10
SMPHAWIMNC
>11
DFRCGTLDVEI
>12
HDMFGDYNYEHD
```

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

```r
lengths <- 6:12
fasta_seqs <- sapply(lengths, function(i) {
  seq <- generate_protein(i)
  paste0(">", i, "\n", paste(seq, collapse = ""))
})
cat(paste(fasta_seqs, collapse = "\n"))
```

```
>6
KVYTES
>7
MVFSCCH
>8
ACEQNPEY
>9
```

```
EYPILCENT
>10
RWQVRHNFPM
>11
IKMMHMEMQAW
>12
VRQMTTIEVHIW
```