Class 6: R functions

Minjun Kang (PID: A69042800)

Table of contents

All functions in R have at least 3 things:

- A **name**, we pick this and use it to call our function,
- Multiple or single input **arguments**
- The **body** lines of R code that do the work

Our first function

Write a function to add some numbers

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

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

```
[1] 110
```

A second function

Write a function to generate random nucleotide sequences of a user specified length

```r
sample(c("A", "C", "G", "T"), size=5, replace=TRUE)
```

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

I want the a 1 element long character vector that looks CACAG...

```r
v <- sample(c("A", "C", "G", "T"), size=5, replace=TRUE)
paste(v, collapse = "")
```

```
[1] "GTGAC"
```

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

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

```r
generate_dna(60, TRUE)
```

```
[1] "TATGCCAAAATTAGACGGCGTTGAGTCCCTAGTGGTGTCGACGAAAACGTGTAGCTAATA"
```

A protein generating function

```r
generate_protein <- function(size = 50, fasta = TRUE) {
  # 20 standard amino acids (single-letter codes)
  amino_acids <- c("A", "R", "N", "D", "C",
                   "E", "Q", "G", "H", "I",
                   "L", "K", "M", "F", "P",
                   "S", "T", "W", "Y", "V")

  # Sample amino acids
  v <- sample(amino_acids, size, replace = TRUE)
```

```
  s <- paste(v, collapse = "")

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

```
generate_protein(50)
```

```
[1] "NWSHHGNNHSFEKEQMNDPAQMAGGFWGKPQCWHKDSQFHQAFWETLSQV"
```

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)

```
length <- 6:12
for (i in length){
  cat(">", i, "\n", sep="")
  a <- generate_protein(i)
  cat(a)
  cat("\n")
}
```

```
>6
SIVQRA
>7
WYMAYDK
>8
NLRFAHQM
>9
VRSWASNEL
>10
GWLMTKKICE
>11
PEPWPPAHPCW
>12
GLQNALEGDRID
```

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

```
sapply(length, generate_protein)
```

```
[1] "CAVVLN"       "FKVPKYT"      "HFNTKKEM"     "LYNNHLPLS"    "NEFDHDNVDL"
[6] "GNWLHWLRLFL"  "STPADGICFNCE"
```