Class 06: Functions

Li Ling (A15092789)

Table of contents

All functions in R have at least 3 things:

- A **name**, we oucj this and use it to call our function,
- Input **arguments** (there can be multiple)
- The **body** lines o 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 the 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 specific length:

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

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

I want the a 1 element long character vector that looks like this "CACAG" not "C" "A" "C" "A" "G".

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

```
[1] "CTCGAGAGGGTGATTCCGGTGCCAGTAGATCGGTAGTGAGGAGAGGCTCG"
```

Turn this into my first weee function!

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

Test it:

```
generate_dna(10)
```

```
[1] "GACATCGGCA"
```

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

```
No you don't!
```

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","T","C","G"), size=size, replace=TRUE)
  paste(v, collapse = "")}
```

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

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

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

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

A Protein generating function

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

if(fasta){
  return(s) #single strings, like FASTA
} else{
  return(v) #multi-element vector
}
}
```

```
generate_protein(10, fasta=FALSE)
```

```
 [1] "M" "Q" "N" "R" "P" "A" "S" "K" "A" "W"
```

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

One way to do this "brute force":

```
generate_protein(10)
```

```
[1] "IATIWNEKWC"
```

```
generate_protein(6)
```

```
[1] "LATITT"
```

```
generate_protein(7)
```

```
[1] "IPCHCKV"
```

```
generate_protein(8)
```

```
[1] "HAWSGPTC"
```

```
generate_protein(9)
```

```
[1] "YCILSGQQT"
```

```
generate_protein(11)
```

```
[1] "RMANIRLYFWE"
```

```
generate_protein(10)
```

```
[1] "DNDRIDLPKI"
```

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

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

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

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

```
>6
AAIMTI
>7
NPHRDST
>8
YIRQTWIE
>9
EQVGQPYSL
>10
AHRRSIHQHA
>11
IWIHMIRTFTW
>12
GVSKNIAEYHLQ
```

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

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

```
[1] "SPDLQV"       "HDSDLKD"      "CKRCMGIS"     "ILWRMSSMT"    "NPPIVGNEGS"
[6] "SEVNHHGCPQL"  "SRRFTRRGMGRV"
```