Class 06: R functions

Libby Gilmore (PID: A69047570)

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

```
# some functions have parameters that are absolutely required, others you can build defaults
add <- function(x, y=1) {
  x + y
}
```

Now we can call this function:

```
add(10) # uses default argument for y, adds 1 by default
```

```
[1] 11
```

```
add(12,7) # sets x and y arguments, and adds them together
```

```
[1] 19
```

```
#add(10,10,100) # Error in add(10, 10, 100) : unused argument (100)

add(c(10,10),100) # alternative to get 100 to add to each element of x vector
```

```
[1] 110 110
```

A second function

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

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

```
# will keep giving you 3 random letters, BUT as replace=FALSE, it'll produce an error if leng
sample(c("A", "C", "G", "T"), size=3)
```

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

```
# with replacement on, you can get a longer sampling
sample(c("A", "C", "G", "T"), size=100, replace=TRUE)
```

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

I want a 1 element long character vector that looks like “CACAGC” and not “C” “A” “C” “A” “G” “C”

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

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

Test it:

```
generate_DNA(60)
```

```
[1] "GCTCTGGTAGCTTTCGTGTCTCGCCCGAGATCAATAATACGAGAGTGGACGCCCAGTGAT"
```

```
# used for flow control
fasta<-FALSE # when TRUE it returns the first clause
if(fasta){
  cat("HELLO You!")
} else {
  cat("No you don't")

}
```

```
No you don't
```

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

```
generate_fasta <- function(size=50, fasta=TRUE){
  nucleotides <- c("A", "C", "G", "T")
  v <- sample(nucleotides, size, replace = TRUE)
  if(fasta){
    return(paste(v, collapse="")) # you can also return without return and just the variable
  } else{
    return(v)
  }
}
```

Test generate_fasta():

```
generate_fasta(50, FALSE)
```

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

```
generate_fasta(20, TRUE)
```

```
[1] "TGCCTAGCACAAGGGTGTAC"
```

A protein generating function

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

```
generate_protein(6)
```

```
[1] "RFYMDM"
```

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)

```
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
MTPLVQ
>7
FQSFQYL
>8
VMTDCFML
```

```
>9
APINGMACA
>10
QEYSKTNLCH
>11
CWPMLQDKIRD
>12
DHVEKPRQPWNM
```

```
# but the `apply` functions are typically more efficient than `for loops`
sapply(lengths, generate_protein)
```

```
[1] "GYNLEC"       "TWPVQGE"      "GPQMDEVW"     "TMGPCYEYS"    "ADKRAWMDNY"
[6] "VNKVVPRAMWG"  "SIWHRSWRTEWC"
```