Class 6: R functions

Barry (PID: 911)

Today we are going to get more exposure to functions in R.

Let's start with a silly simple function to add some numbers:

```
add <- function(x, y=0, z=0) {
  x + y + z
}
```

Can we use this function

```
add(1,1)
```

```
[1] 2
```

```
add(c(100,200), 1)
```

```
[1] 101 201
```

```
add(x=100, y=1)
```

```
[1] 101
```

```
add(100)
```

```
[1] 100
```

```
log(10)
```

```
[1] 2.302585
```

```
log(10, base=10)
```

```
[1] 1
```

```
add(100, 1, 200)
```

```
[1] 301
```

A more intresting example

Let's have a look at the `sample()` function.

Q. What does it do

The `sample()` function in R randomly selects elements from a vector. It has two main uses:

```
sample(1:10, size=5)
```

```
[1]  4 10  7  8  2
```

```
sample(1:10, size=1)
```

```
[1] 9
```

What if I want 11 things taken from my vector 1 to 10

```
sample(1:10, size=11, replace=T)
```

```
 [1]  8 10  4  3  3  7  2  3  4  9  2
```

```
x <- 1:10
x
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

Side-note:

```
seq(5, 50, by=3)
```

```
 [1]  5  8 11 14 17 20 23 26 29 32 35 38 41 44 47 50
```

Generate DNA sequences

> Q. Write a function to generate a random nucleotide sequence of a user specified size/length.

```
x <- c("A", "C", "G", "T")
sample(x, size=9, replace=T)
```

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

All function in R have at least 3 things:

- a **name** (we pick this "generate_dna")
- input **arguments** ("length" of the output sequence)
- the **body** (where the work gets done, line by line)

```
generate_dna <- function(length=10) {
  bases <- c("A","C","G","T")
  ans <- sample(bases, size=length, replace=TRUE)
  return(ans)
}
```

```
s <- generate_dna()
s
```

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

```
s <- generate_dna(40)
s
```

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

I would like my function to print out a single element vector "GATGATCT". To help with this I can maybe use the `paste()` function.

```
s
```

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

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

```
[1] "ATTGGGAGTACTTGTCGGCATGGGACCCTTAGGTTGCATA"
```

```
generate_dna <- function(length=10) {
  # The nucleotides to draw/sample from
  bases <- c("A","C","G","T")
  # Draw n=length nucleotides to make our sequence
  ans <- sample(bases, size=length, replace=TRUE)
  # Concatenate/join/paste sequence into one word
  ans <- paste(ans, collapse="")
  return(ans)
}
```

```
s <- generate_dna(length=9)
s
```

```
[1] "ACGACTCGA"
```

I want the ability to switch between these two output formats. I can do this with an extra input argument to my function that controls this with TRUE/FALSE

```
generate_dna <- function(length=10, collapse=FALSE) {
  # The nucleotides to draw/sample from
  bases <- c("A","C","G","T")
  # Draw n=length nucleotides to make our sequence
  ans <- sample(bases, size=length, replace=TRUE)

  # Concatenate/join/paste sequence into one word
  if(collapse) {
    ans <- paste(ans, collapse="")
```

```
  }
  return(ans)
}
```

```
generate_dna(length=5, collapse=TRUE)
```

```
[1] "CAATC"
```

Q. Add the ability to print a wee msg if the user is sad. Control this with a new input paramater called `mood`.

```
cat("Hellllooooo")
```

```
Hellllooooo
```

```
generate_dna <- function(length=10, collapse=FALSE, mood=FALSE) {
  # The nucleotides to draw/sample from
  bases <- c("A","C","G","T")
  # Draw n=length nucleotides to make our sequence
  ans <- sample(bases, size=length, replace=TRUE)

  # Concatenate/join/paste sequence into one word
  if(collapse) {
    ans <- paste(ans, collapse="")
  }

  if(mood) {
    cat("Cheer up we are nearly done!\n")
  }
  return(ans)
}
```

```
generate_dna(4, mood=T)
```

```
Cheer up we are nearly done!
```

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

Q. Write a protein sequence generating function with the ability to output random amino acid sequences of a user defined length.

```
aa <- c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I", "L", "K", "M", "F", "P", "S", "T",

length(aa)
```

```
[1] 20
```

```
generate_protein <- function(length=10, collapse=FALSE) {
  # The nucleotides to draw/sample from
  aa <- c("A", "R", "N", "D", "C", "Q", "E",
          "G", "H", "I", "L", "K", "M", "F",
          "P", "S", "T", "W", "Y", "V")
  # Draw n=length nucleotides to make our sequence
  ans <- sample(aa, size=length, replace=TRUE)
  # Concatenate/join/paste sequence into one word
  if(collapse) {
    ans <- paste(ans, collapse="")
  }
  return(ans)
}
```

```
generate_protein(6)
```

```
[1] "G" "H" "G" "Y" "L" "C"
```

```
generate_protein(6, TRUE)
```

```
[1] "MGRPPL"
```

Q. Generate protein sequences from length 6 to 12 amino acids long.

```
#generate_protein(6:12, TRUE)
```

This does not work because my function is not vectorized (in other words, setup to work on each element of the first input argument `length`).

In particular, we can use `sapply()` to solve this.

Or brute force…. copy/paste…

```
generate_protein(6, TRUE)
```

```
[1] "NKGIKD"
```

```
generate_protein(7, TRUE)
```

```
[1] "PMHNQFG"
```

```
generate_protein(8, TRUE)
```

```
[1] "SDMKIYDG"
```

```
generate_protein(9, TRUE)
```

```
[1] "ICEDHGTKI"
```

The `sapply()` function applies a function to each element of a vector/list and simplifies the output

```
sapply(6:12, generate_protein, collapse=T)
```

```
[1] "MVKCWM"       "CWPILDE"      "FFISCSRK"     "FWAYCQACC"    "EEWWPQVNHV"
[6] "QAFMIWCQGHQ"  "LDKKKNAQFYII"
```

> Q. Are any of these sequences unique in the sense that they have never been found in nature?

To make this accessible lets get our sequences in FATA format.

FASTA format looks this

> id.6 GTAGKRLP id.7 KRTYFREGG

```
myseqs <- sapply(6:12, generate_protein, collapse=T)
myseqs
```

```
[1] "PRKAHE"       "GKSKLYK"      "AHDLGEPH"     "FIVDEGHNG"    "SSPQVKRGVI"
[6] "RFMQNYGMVVH"  "AMHYSWVQTCED"
```

The functions`paste()` and `cat()` will help here

```
cat( paste(">id.", 6:12, "\n", myseqs, "\n", sep=""), sep="" )
```

```
>id.6
PRKAHE
>id.7
GKSKLYK
>id.8
AHDLGEPH
>id.9
FIVDEGHNG
>id.10
SSPQVKRGVI
>id.11
RFMQNYGMVVH
>id.12
AMHYSWVQTCED
```

```
library(bio3d)

myseqs.vec <- sapply(6:12, generate_protein, collapse=T)
x <- as.matrix(myseqs.vec)
x
```

```
     [,1]
[1,] "RQQQPY"
[2,] "DLLDMHG"
[3,] "YDWRGRFS"
[4,] "QWWAYQMAI"
[5,] "THISIAECIL"
[6,] "VHPSHPNKTKQ"
[7,] "DHKWRNWKSPMT"
```