Class 6: R functions

Katie Mostoller (A17259578)

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

```
[1] 2
```

```
add(x=1)
```

```
[1] 1
```

```
log(10)
```

```
[1] 2.302585
```

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

```
[1] 1
```

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

```
[1] 301
```

Let's have a look at the `sample()` function. >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]  3 10  8  4  5
```

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

```
[1] 2
```

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

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

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

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

> 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] "C" "G" "G" "T" "C" "A" "A" "G" "A"
```

All functions 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) {x <- c("A", "C", "G", "T")
ans <- sample(x, size=length, replace = T)}
```

```
generate_dna(12)
```

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

```
s <- generate_dna(10)
paste(s, collapse = "")
```

```
[1] "AGTTTGAGCG"
```

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

```
generate_dna(12)
```

```
[1] "CCTTTCCAGTGA"
```

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, collapse=FALSE) {
# The nucleotides to draw/sample from
x <- c("A", "C", "G", "T")
# Draw n=length nucleotides to make our sequence
ans <- sample(x, size=length, replace = T)
# Concatenate the nucleotides
if(collapse) {
ans <- paste(ans, collapse = "")
}
return(ans)}
```

```
generate_dna(12)
```

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

```
generate_dna(12, collapse = TRUE)
```

```
[1] "GCTTTTAAGAGA"
```

Add the ability to put a message if the use is sad. Control this with a new input parameter called `mood`

```
generate_dna <- function(length, collapse=FALSE, mood=FALSE) {
# The nucleotides to draw/sample from
x <- c("A", "C", "G", "T")
# Draw n=length nucleotides to make our sequence
ans <- sample(x, size=length, replace = T)
# Concatenate the nucleotides
if(collapse) {
ans <- paste(ans, collapse = "")
}
#
if(mood) {
cat("Cheer up we are nearly done")
}
return(ans)}
```

```
generate_dna(12, collapse=TRUE, mood=TRUE)
```

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

```
[1] "GTACTGGAATTG"
```

Write a protein sequence generating function

```
generate_protein <- function(length, collapse = TRUE) {
# amino acids to draw from
aa_codes <- c("A", "C", "D", "E", "F",
              "G", "H", "I", "K", "L",
              "M", "N", "P", "Q", "R",
              "S", "T", "V", "W", "Y")
```

```
# Draw n=length amino acids to make our sequences
ans <- sample(aa_codes, size = length, replace = T)
# Concatenate amino acids
if(collapse) {
ans <- paste(ans, collapse = "")
}
return(ans)
}
```

```
generate_protein(12)
```

```
[1] "SGHMEIQNTRAH"
```

Generate protein sequences from length 6-12 amino acids long

```
generate_protein(6)
```

```
[1] "LRKKSV"
```

```
generate_protein(7)
```

```
[1] "WSKMRLS"
```

```
generate_protein(8)
```

```
[1] "HMMVCMMW"
```

```
generate_protein(9)
```

```
[1] "NHSIILKTV"
```

```
generate_protein(10)
```

```
[1] "YFYMVQAVGQ"
```

```
generate_protein(11)
```

```
[1] "RNIIMKRGRWK"
```

```
generate_protein(12)
```

```
[1] "LEIIIGHMDATV"
```

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

This does not work because my function is not vectorized! We can use `sapply()` to fix this. 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] "CLGYTE"       "RCPPVDG"      "DMLQFIIP"     "GDNKCVFGN"    "VKWRYRAAWQ"
[6] "RRGVFYPKCAD"  "STGYYFSWGMHI"
```

> 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 FASTA format. FASTA format looks like this: >id.6 GTAGKRLP >id.7 KRTYFREGG

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

```
[1] "SLQNGA"       "QMCKGPH"      "RLQDRMKR"     "TNAGLQPRQ"    "DHQGNKWSTH"
[6] "DVYLSEHRNCV"  "GPSIAWVYAPRL"
```

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

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

```
>id.6
SLQNGA
>id.7
QMCKGPH
>id.8
RLQDRMKR
>id.9
TNAGLQPRQ
>id.10
DHQGNKWSTH
>id.11
DVYLSEHRNCV
>id.12
GPSIAWVYAPRL
```

```
library(bio3d)

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

x <- as.matrix(myseqs.vec)

x
```

```
     [,1]
[1,] "RAGWLD"
[2,] "MITIQIM"
[3,] "FFARCHRH"
[4,] "KAADTKVQI"
[5,] "PLYDQMGPDQ"
[6,] "CHSTLCTFVML"
[7,] "TYLKWGPPLFEQ"
```

Using a blast search, the amino acid sequences start being unique at length 10.