Class 6: R functions

Yiyu

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

Let's start with a silly simple function that adds some numbers:

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

Can we use this function?

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

```
[1] 2
```

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

```
[1] 101 201
```

```r
add(100)
```

```
[1] 100
```

```r
log(10)
```

```
[1] 2.302585
```

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

```
[1] 1
```

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

```
[1] 301
```

##A more interesting example

Let's have a look at `sample()`

> Q. What does it do?

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

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

```
[1] 3 4 5 7 1
```

What if I want 11 things drawn from my vector 1 to 10? Must set replace to TRUE to allow for repetition.

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

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

###Generate DNA sequences

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

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

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

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)

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

```r
generate_dna()
```

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

```r
s <- generate_dna()

s
```

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

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

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

```
[1] "CACGCCGTCT"
```

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

```r
generate_dna(20)
```

```
[1] "AATATAAGGGGAGGAAAATA"
```

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

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

```r
generate_dna(4)
```

```
[1] "CCCG"
```

> Q. Add the ability to print a message depending on user mood. Control this with a new input parameter called `mood`.

```r
generate_dna <- function(length=10, collapse=TRUE, mood= FALSE) {
  bases <- c("A", "G", "C", "T")
  ans <- sample(bases, size=length, replace = T)
  if(collapse){
   ans <- paste(ans, collapse = "")
  }
  if(mood){
    cat("yipee")
  }
  return(ans)
}
```

```r
generate_dna(9, mood = T)
```

```
yipee
```

```
[1] "AAGCGCGTC"
```

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

```r
generate_protein <- function(length=10, collapse = TRUE){
  amino_acids <- c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I",
                   "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V")
  ans <- sample(amino_acids, size=length, replace = T)
  if(collapse){
     ans <- paste(ans, collapse = "")
  }
  return(ans)
}
```

```r
generate_protein(6)
```

```
[1] "KYGWRH"
```

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

```r
generate_protein(6)
```

```
[1] "MKTGWC"
```

```r
generate_protein(7)
```

```
[1] "TPYENDH"
```

```r
generate_protein(8)
```

```
[1] "MFCRRICN"
```

```r
generate_protein(9)
```

```
[1] "PKRLRTTVS"
```

```r
generate_protein(10)
```

```
[1] "MQDDDSRNVR"
```

```r
generate_protein(11)
```

```
[1] "QQDALKIPTQH"
```

```r
generate_protein(12)
```

```
[1] "ITRDEINHVMQS"
```

> Q. How do I do this more efficiently?

Could use `sapply()` function. It applies a function to each element of a vector/list and simplifies the output.

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

myseqs
```

```
[1] "SVMMME"        "SKNTMPP"       "PDCEVLIM"      "VAIIRMKSP"     "PREREADSWW"   
[6] "KDSPLDARMIF"   "YFSRHESFYDKN"
```

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

To make this accessible, let's get our sequences in FASTA format.

FASTA format looks like this

> id.6 GTAGKRL id.7 KRTYFREGG

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

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

```
>id.6
SVMMME
>id.7
SKNTMPP
>id.8
PDCEVLIM
>id.9
VAIIRMKSP
```

```
>id.10
PREREADSWW
>id.11
KDSPLDARMIF
>id.12
YFSRHESFYDKN
```