R functions

Jessica Gao (PID A16939806)

Today we will get more exposure to functions in R. We call functions to do all our work and today we will learn how to write our own.

##A first silly function

Note that arguments 2 and 3 have default values (because we set y=0 and z=0) so we son't have to supply them when we call our function).

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

#0 is being set as default for y, if you don't give function a y
```

Can I just use this

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

```
[1] 2
```

```
add(1,c(10,100))
```

```
[1]  11 101
```

```
#could do add(x=1, y=c(10,100)), without labels is fine too
```

```
add(100)
```

```
[1] 100
```

```
add(100,10,1)
```

```
[1] 111
```

A second more fun function

Let’s write a function that generates random nucleotide sequences.

We can make use of the in-built ‘sample()’ function in R to help us here.

```
sample(1:10, size=9)
```

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

```
#size=9 means program gives 9 outputs
#size cannot be larger than the size of population, because the default replace= is false.

sample(1:10, size=9, replace=TRUE)
```

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

> Q. Can you use ‘sample()’ to generate a random nucleotide sequence of length 5

```
#save nucleotide as vector
nucleotide <- c("A", "T", "C", "G")
sample(nucleotide, size=5, replace=TRUE)
```

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

```
#OR
sample(x=c('A','T','C','G'), size=5, replace=T)
```

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

> Q. Write a function ‘generate_dna()’ that makes a nucleotide sequence of a user specified length.

Every function in R has at least 3 things”

- a **name** (in our case ‘generate_dna’)

- one or more **input arguments** (the 'length' of sequence we want)
- a **body** (that does the work)

```
generate_dna<-function(length=5){
  bases<- c('A','C','T','G')
  sample(bases, size=length, replace=TRUE)
}

#length=5 is setting the default
```

```
generate_dna(10)
```

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

```
generate_dna(100)
```

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

```
#install.packages('bio3d')
library(bio3d)
bio3d::aa.table$aa1[1:20]
```

```
 [1] "A" "R" "N" "D" "C" "Q" "E" "G" "H" "I" "L" "K" "M" "F" "P" "S" "T" "W" "Y"
[20] "V"
```

> Q.Can you write a 'generate_protein()' function that returns amino acid sequence of a user requested length?

```
generate_protein <- function(length=5) {
  amino_acids <- bio3d::aa.table$aa1[1:20]
  sample(amino_acids, size=length, replace=T)
}
```

```r
generate_protein(30)
```

```
 [1] "H" "H" "I" "Y" "C" "R" "H" "W" "W" "F" "N" "W" "T" "T" "I" "G" "R" "R" "P"
[20] "W" "V" "H" "R" "G" "F" "W" "D" "F" "E" "S"
```

I want my output of this function not to be a vector with one amino acid per element but rather a one element single string

```r
bases <- c('A','G','T','C')
paste(bases, collapse ='-----')
```

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

```r
#collapse= puts whatever you set it equal to be in between the elements, paste() gives one s
```

```r
generate_protein <- function(length=5) {
  amino_acids <- bio3d::aa.table$aa1[1:20]
  s <- sample(amino_acids, size=length, replace=T)
  paste(s, collapse= '')
}
```

```r
generate_protein()
```

```
[1] "YFKCD"
```

> Q. Generate protein sequences from length 6 to 12.

```r
generate_protein(length=6)
```

```
[1] "CQDMCP"
```

```r
generate_protein(length=7)
```

```
[1] "HRSSHYV"
```

```r
generate_protein(length=8)
```

```
[1] "PWYNICTS"
```

We can use the useful utility function 'sapply()' to help us "apply" our function over all the values 6 to 12

```r
ans <- sapply(6:12, generate_protein)
#apply the function multiple times to 6:12
ans
```

```
[1] "AVTVWW"       "EDVVRMS"      "NAGHCCFM"     "LFFCPDKSP"    "NTRYVWSQTG"
[6] "SPRTILLQPGK"  "MWCMCAVLTQHE"
```

```r
#creating FAFSTA format
cat(paste('>ID.', 6:12, sep='', '\n', ans, '\n'), sep="")
```

```
>ID.6
AVTVWW
>ID.7
EDVVRMS
>ID.8
NAGHCCFM
>ID.9
LFFCPDKSP
>ID.10
NTRYVWSQTG
>ID.11
SPRTILLQPGK
>ID.12
MWCMCAVLTQHE
```

> Q. Are any of these sequences unique in nature - i.e. never found in nature. We can search "refseq-protein" and look for 100% identity and 100% coverage.

All of these sequences are not unique in nature. They all have 100% identity, 100% coverage after blasting.