Class06

Ivan Henry Kish(PID:A17262923)

Table of contents

Background

Functions are at te heart of using R. Everything we do involves calling and using functions (from data input, analysis to results output).

All functions in R have at least 3 things:

1. A **name** the thing we used to called the function.
2. One or more input **arguments** that are comma separated
3. The **body**, lines of code between curly brackets {} that does the work of the function.

A first function

Let's write a silly wee function to add some numbers:

```
add <- function(x) {
  x + 1
}
```

Let's try it out

```
add(100)
```

```
[1] 101
```

Now will this work?

```
add(c(100,200,300))
```

```
[1] 101 201 301
```

I am going to modify the function to be more useful.

```
add <- function(x,y=1) {
  x + y
}
```

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

```
[1] 110
```

```
add(100)
```

```
[1] 101
```

> **N.I.K.** Input arguments can be either **required** or **optional**. The later have a fall-back efault that is specified in the function coed with an = sign.

```
#add(100,200,300)
```

A 2nd function

All functions in R look like this

```
name <- function(arg){
body
}
```

The `sample()` function in R selects random values at a specified number of random selections. You can also specify whether or not you want values to be repeatedly picked or not.

```
sample(1:10, size=4, replace=FALSE)
```

```
[1] 10  7  4  1
```

Q. Return 12 numbers picked randomly from the input 1:10

```
sample(1:10, size=12, replace=TRUE)
```

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

Q. Write the code to generate a random 12 nucleotide log DNA sequence

```
sample(c("A","T","G","C"), size=12, replace=TRUE)
```

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

Q. Write a first. version function called `generte_dna()` that generates a user specified length `n` random DNA sequnce.

```
generate_dna<- function(n){
  sample(c("A","T","G","C"), size=n, replace=TRUE)
}
generate_dna(26)
```

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

```
generate_dna(45)
```

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

Q. Modify our function to return a FASTA like sequence rather than the quoted nucleotides we have been getting.

```
generate_dna<- function(n){
  seq<-sample(c("A","T","G","C"), size=n, replace=TRUE,)
  paste(seq,collapse = "")
}
```

```
generate_dna(45)
```

```
[1] "TCAAAAAAATATCCGATAGAGCCCCAGACATTTTCTGTTCTTCCT"
```

Q. Give the user an option to return FASTA format output sequence or standard multi-element vector format.

```
generate_dna<- function(n, fasta=TRUE){
  seq<-sample(c("A","T","G","C"), size=n, replace=TRUE,)

  if(fasta){
  seq<-paste(seq,collapse = "")
  }
  return(seq)
}
```

```
generate_dna(12,fasta=T)
```

```
[1] "CAAATCGTCCGG"
```

A new cool function

Q. Write a function called `generate_protein()` that generates a user specified length protein sequence in FASTA like format?

```
generate_protein<- function(n, fasta=TRUE){
  seq<-sample(c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I",
        "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V"), size=n, replace=TRUE,)

  if(fasta){
  seq<-paste(seq,collapse = "")
  }
  return(seq)
}
```

```
generate_protein(12)
```

```
[1] "AYMMVMAASMEH"
```

Q. Use your new `generate_protein()` function to generate sequences between length 6 and 12 amino acids in length and check any of these are unique in nature (found in the NR database at NCBI).

```
generate_protein<- function(n, fasta=TRUE){
  seq<-sample(c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I",
        "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V"), size=n, replace=TRUE,)

  if(fasta){
  seq<-paste(seq,collapse = "")
  }
  return(seq)
}
```

```
for(i in 6:12){
  cat(">", i,sep="", "\n")
  cat(generate_protein(i),"\n")
}
```

```
>6
LPNHTG
>7
SEMCPDN
>8
LLGTCADY
>9
SAQNWIKWL
>10
VWWDIWYGYP
>11
MHEATNYEAPG
>12
QEKWERFMDCTD
```

Identical matches were only seen in the amino acids that were generated with lengths 6 and 7.