class06: R Functions

Kenny (PID: 18544481)

Table of contents

Background

Functions are at the 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:

- A **name** the thing we use to call the function.
- One or more input **arguments** that are comma separated
- 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(103)
```

```
[1] 104
```

Will this work

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

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

Modify to be more useful and add more than just 1

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

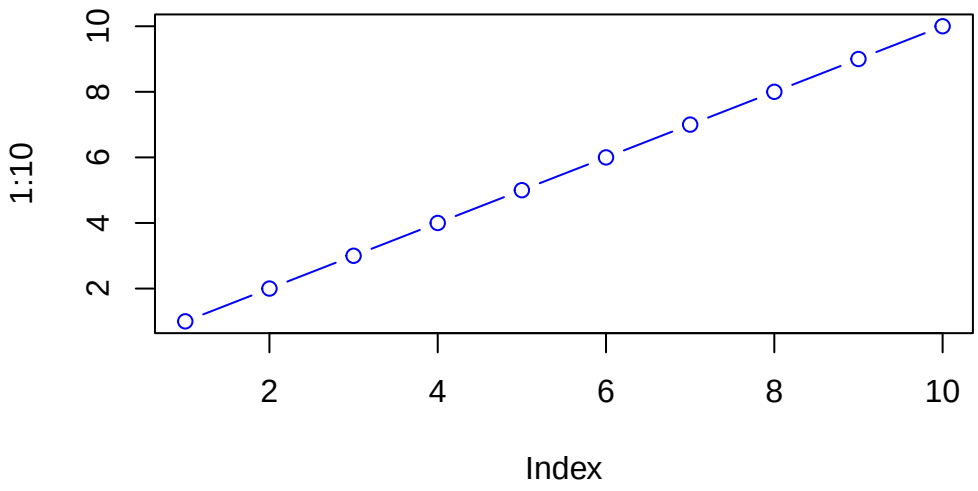
```
add (100,10)
```

```
[1] 110
```

Will this work?

```
add <- function (x) {
  100
  }
```

```
plot (1:10, col= "blue", typ = "b")
```




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

```
[1] 1
```

```
add(100)
```

```
[1] 100
```

> **N.B.** Input arguments can be either **required** or **optional**. The later have a fall-back default that is specified in the function code with an equals sign.

```
#add (x=100, y=200, z=300)
```

A second function

All functions in R look like this

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

The `sample()` function in R takes a sample of the specified size from the element of x using either with or without replacement. Essentially, it randomly selects items from a vector.

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

```
[1] 9 4 5 6 2
```

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

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

```
 [1]  9  1  5  7  2  7 10  1  6  7  1  2
```

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

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

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

Q. Write a first version function called `generate_dna()` that generates a user specified length `n` random DNA sequence?

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

```
generate_dna <- function(n=6) {
  bases <- c ("A", "T", "C", "G")
  sample (bases, size=n, replace=TRUE)
}
```

```
generate_dna(100)
```

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

Q. Modify your function to return a FASTA like sequence so rather than [1] “G” “C” “A” “A” “t” we want “GCAAT”

```
generate_dna <- function(n=6) {
  S1 <- sample (c("A", "T", "C", "G"), size=n, replace=TRUE)
  paste (S1, collapse = "")
}
generate_dna(5)
```

```
[1] "TTGCC"
```

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

```
generate_dna <- function (n=6, fasta=TRUE) {
bases <- c("A", "C", "G", "T")
ans<- sample (bases, size=n, replace =TRUE)

if(fasta) {
    ans <- paste (ans, collapse = "")
    cat("Hello...")
} else {
  cat ("... is it me you are looking for")
}
return(ans)
}
```

```
generate_dna(10)
```

```
Hello...
```

```
[1] "TGGCTTTACG"
```

```
generate_dna(10, fasta=F)
```

```
... is it me you are looking for
```

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

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=7) {
  aa <- 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)
  paste (aa, collapse = "")
}
generate_protein(7)
```

```
[1] "VYWPLDD"
```

generate_dna <- function(n=6) { S1 <- sample (c("A", "T", "C", "G"), size=n, replace=TRUE) paste (S1, collapse = " ") } generate_dna(5)

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

```
generate_protein <- function(n=7) {
  aa <- 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)
  paste (aa, collapse = "")
}

generate_protein(6)
```

```
[1] "HYIAQP"
```

or we could do a `for()` loop:

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

```
>6
LNVDTG
>7
DRVYLID
>8
VPVARDRW
>9
QFRFKPNNH
>10
ITANDYTQSL
>11
WWQSSARMEQP
>12
KYLDEIKYCQCM
```