

Class06: R Functions

Melanie Alonzo (A17375327)

Table of contents

Background	1
A first function	1
A second function	3
A new cool function	6

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:

1. A **name** the thing we use to call the function.
2. One or more input **arguments** that are comma seperated
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
```

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)
```

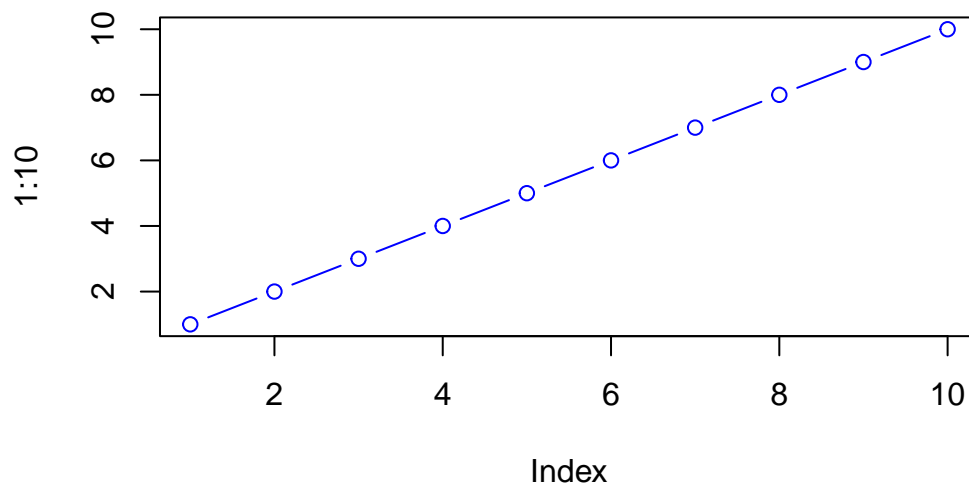
```
[1] 110
```

Will this work?

```
add(100)
```

```
[1] 101
```

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



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

```
[1] 1
```

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(100, 200, 300)
```

A second function

All functions in R look like this

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

The `sample()` function in R...

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

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

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

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

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

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

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

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

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) {  
  sample(c("A","C", "G", "T"), size= n, replace= TRUE)  
}
```

```
generate_dna(100)
```

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

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

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

```
generate_dna(10)
```

```
[1] "CTGTAGTCGG"
```

An Example

```
# Example pattern (not using your bases)
x <- c("H", "E","L", "L", "O")
paste(x, collapse = "****")
```

```
[1] "H***E***L***L***O"
```

```
#returns "HELLO"
```

```
n <- c("A","C", "G", "T")
paste(n, collapse = "")
```

```
[1] "ACGT"
```

Q. Give the user and option to return FASTA format output 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] "ATTGAATGA"
```

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

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

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

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=6, fasta= TRUE) {  
  aa <- c("A","R","N","D","C",  
          "Q","E","G","H","I",  
          "L","K","M","F","P",  
          "S","T","W","Y","V")  
  ans <- sample(aa, size=n, replace= TRUE)  
  
  if (fasta) {  
    ans <- paste(ans, collapse="")  
  }  
  return(ans)  
}
```

```
generate_protein(10)
```

```
[1] "EWIQWELMWH"
```

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

```
[1] "F" "K" "W" "P" "V" "W" "K" "K" "G" "F"
```

Q. Use your new `generate_protein()` function to generate all 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(6)
```

```
[1] "GIVYAN"
```

```
generate_protein(7)
```

```
[1] "AIQLGHP"
```

```
generate_protein(8)
```

```
[1] "RGQESVLE"
```

```
generate_protein(9)
```

```
[1] "WHGCNWEAY"
```

```
generate_protein(10)
```

```
[1] "ESHDPGDRIH"
```

```
generate_protein(11)
```

```
[1] "QTCHSNLGDVD"
```

```
generate_protein(12)
```

```
[1] "HRHFNNMQIGMD"
```

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

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

```
>6  
QSRVEI  
>7  
GDMHWFA  
>8  
KMDWLEFE  
>9  
DFGGSEFPW  
>10  
NRVKHIRAID  
>11  
MPVNKKMLCIT  
>12  
MYSIPMPHNDEC
```