

# Class 6: R Function

Wade Ingersoll (PID: A69038080)

## Table of contents

Our first (silly) function . . . . .	1
A second function . . . . .	2
Protein generating function . . . . .	3

All functions in R have at least 3 things:

- A **name**, we pick this and use it to call our function,
- Input **arguments** (there can be multiple)
- The **body** lines of R code that do the work

## Our first (silly) function

Write a function to add some numbers

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

Now we can call this function:

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

```
[1] 110 110
```

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

```
[1] 110
```

## A second function

Write a function to generate random nucleotide sequences of a user specified length:

The `sample()` function can be helpful here.

```
sample(c("A", "C", "G", "T"), size=50, replace=TRUE)
```

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

I want a 1-element long character vector that looks like: “CCAGTC”, not “C” “C” “A” “G” “T” “C”

```
v <- sample(c("A", "C", "G", "T"), size=50, replace=TRUE)  
paste(v, collapse="")
```

```
[1] "CCAGTATCGCCTTGCTGATGGTGTAGGCAGCACTTCAAGCTCGCTCCC"
```

Turn this into my first function

```
generate_dna <- function(size = 50) {  
  v <- sample(c("A", "C", "G", "T"), size = size, replace = TRUE)  
  paste(v, collapse = "")  
}
```

Test it:

```
generate_dna(30)
```

```
[1] "AATTATCCCGAAATGTTCTGGGGCACTGA"
```

Practicing IF:

```
fasta <- TRUE  
if(fasta) {  
  cat("HELLO You!")  
} else {  
  cat("No you don't!")  
}
```

HELLO You!

Add the ability to return a multi-element vector or a single element fasta-like element:

```
generate_dna <- function(size = 50, fasta=TRUE) {  
  v <- sample(c("A", "C", "G", "T"), size = size, replace = TRUE)  
  s <- paste(v, collapse = "")  
  
  if(fasta){  
    return(s)  
  } else {  
    return(v)  
  }  
}
```

Try it:

```
generate_dna(fasta = TRUE)
```

```
[1] "GAGAGTAGGTAGATATAAAGACAGCCGAATAGACGTGCTTATGCCGAA"
```

```
generate_dna(fasta = FALSE)
```

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

## Protein generating function

```
generate_protein <- function(size = 50, fasta = TRUE) {  
  aa <- c("A", "R", "N", "D", "C", "Q", "E", "G",  
         "H", "I", "L", "K", "M", "F", "P", "S",  
         "T", "W", "Y", "V")  
  
  v <- sample(aa, size = size, replace = TRUE)  
  s <- paste(v, collapse = "")  
  
  if (fasta) {
```

```
        return(s)
    } else {
        return(v)
    }
}
```

Try it:

```
generate_protein(10)
```

```
[1] "EHYEAWCHQL"
```

Use our new `generate_protein()` function to make random protein sequences of length 6 to 12 (i.e. one length 6, one length 7, etc. up to length 12)

One way to do this is “brute force”

```
generate_protein(6)
```

```
[1] "NADLWI"
```

```
generate_protein(7)
```

```
[1] "NKVMMIY"
```

```
generate_protein(8)
```

```
[1] "SALRQFLN"
```

```
generate_protein(9)
```

```
[1] "TYNFKSCNE"
```

A second way is to use a `for()` loop:

```
lengths <- 6:12
lengths
```

```
[1] 6 7 8 9 10 11 12
```

```
for(i in lengths) {  
  cat(">", i, '\n', sep="")  
  aa <- generate_protein(i)  
  cat(aa)  
  cat('\n')  
}
```

```
>6  
HWDGEW  
>7  
IAIHNAG  
>8  
AKIWYVSA  
>9  
GTLNKRQST  
>10  
FFMLSKMLLE  
>11  
AMYIWKPLLRL  
>12  
AIKCHLMWLEQF
```

Try `sapply()` function

```
sapply(6:12, generate_protein)
```

```
[1] "ETVVCT"          "DHPNHAG"         "FNKIRFAA"        "DLWVLGDNC"       "ADPSDQHMTK"  
[6] "HSIKYPCGAYC"    "VCMVFDEDILD"
```