

Class 6: R Functions

Qingyun Zheng (PID:A16338254)

Table of contents

Our first (silly) function	1
Our second function	2
A protein generating function	4

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  
}
```

Note: The $y=1$ means that if we don't provide a value for y , it will default to 1.

Now we call this function:

```
add(5,10)
```

```
[1] 15
```

```
add(5)
```

```
[1] 6
```

Input values as vectors

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

```
[1] 11 11 101
```

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

```
[1] 110 110
```

Our 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", "T", "G", "C"), size = 50, replace = TRUE)
```

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

*Replace = TRUE fixes the size issue

I want the a 1 element long character vector that looks have no gaps like "ATCGCTA"

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

```
[1] "TACTAATAGGCCTGTACGGGTATAGGTGCCTTCACCCTAACCCAAGTCA"
```

*The `paste(x, collapse = "")` function removes the gap

Turn the above into a function that takes input length and generates nucleotide sequence of that input length

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

Callling the function above

```
generate_DNA(20)
```

```
[1] "AGTATTGTCCAGTTAACGCA"
```

The if statement has the following structure:

```
if(TRUE){  
  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 vector

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

```
generate_fasta(10, fasta = FALSE)
```

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

```
generate_fasta()
```

```
[1] "CCCTGGGCTAACCTTATGGTACCAAACCTCGGCCCTCGGATTGAGTGAAA"
```

```
generate_fasta(10)
```

```
[1] "CCAATAGCTC"
```

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)  
  }  
}
```

```
generate_protein(60)
```

```
[1] "IGPYINRHVSAKNIQYYDFFMPTTVHQNDWLNDALKEYDGWKKRNLKWCKISQVFHNHF"
```

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

Using `for` loop

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

```
>6  
PASGPR  
>7  
HWFWVGA  
>8  
ICVDRKQR  
>9  
HVAEPIKDD  
>10  
KQKNCQAALP  
>11  
DNACTPITAPY  
>12  
SMNPWATPWAHK
```

```
sapply(6:10, generate_protein)
```

```
[1] "YLIKIH"      "PTHCKAG"     "EFISRIYF"     "SYSREHIWE"   "YVFRQNMKCH"
```