

Class 6: R Functions

Merna Behnam (PID: A17675170)

Background

All functions in R have at least 3 things:

- A **name** that we use to call the function
- One or more input **arguments**
- The **body**, the lines of R code that do the work

Our First Function

Let's write a silly wee function called `add()` to add some numbers (the input arguments)

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

Now we can use this function!

```
add(100,1)
```

```
[1] 101
```

```
add(x=10, y=10)
```

```
[1] 20
```

```
add(x=c(100,1,100), y=1)
```

```
[1] 101    2 101
```

Q. What if I give a mutiple element vector to `x` and `y`?

```
add(x=c(100,1), y=c(100,1))
```

```
[1] 200    2
```

Q. What if I give three inputs to the function?

```
#add(x=c(100,1), y=1, z=1)
```

Q. What if I give only one input to the `add` function?

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

```
addnew(x=100)
```

```
[1] 101
```

```
addnew(c(100,1), 100)
```

```
[1] 200 101
```

If we write our function with input arguments having no default value, then the user will be required to set them when they use the function. We can give our input arguments “default” values by setting them equal to some sensible value - e.g., `y=1` in the `addnew()` function.

A Second Function

Let's try something more interesting: Make a sequence generating tool..

The `sample()` function can be a useful starting point here:

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

```
[1] 1 4 9 5
```

Q. Generate 9 random numbers taken from the input vector `x=1:10`?

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

```
[1] 6 7 5 2 9 5 9 9 3 3 1 8
```

Q. Write code for the `sample()` function that generates nucleotide sequences of length 6?

```
sample(x=c("A","C","G","T"), size=6, replace=TRUE)
```

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

Q. Write a first function `generate_dna()` that returns a *user specified length* DNA sequence:

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

```
generate_dna(len=100)
```

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

Key Points Every function in R looks fundamentally the same in terms of its structure. Basically 3 things: name, input, and body.

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

Functions can have multiple inputs. These can be **required** arguments or **optional** arguments. With optional arguments having a set default value.

Q. Modify and improve our `generate_dna()` function to return its generated sequence in a more standard format like “AGTAGTA” rather than the vector “A”, “C”, “G”, “T”.

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

```
generate_dna <- function(len=6, fasta=TRUE) {

  ans <- sample(x=c("A","C","G","T"),
               size=len, replace = TRUE)

  if(fasta) {
    cat("Multi-element vector output")
    ans <- paste(ans, collapse = "")
  } else {
    cat("Multi-element vector output")
  }
  return(ans)
}
generate_dna(fasta=TRUE)
```

Multi-element vector output

```
[1] "AGATCA"
```

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

Multi-element vector output

```
[1] "TCGTAT"
```

The `paste()` function - it's job is to join up or stick together (a.k.a paste) input strings together.

```
paste("alice", "loves R", sep =)
```

```
[1] "alice loves R"
```

Flow control means where the R brain goes in your code.

```
good_mood <- TRUE

if(good_mood) {
  cat("Great!")
} else {
  cat("Bummer!")
}
```

Great!

Protein Generating Function

Q. Write a function, called `generate_protein()`, that generates a user specified length of protein sequence.

Q. Use that function to generate some different sequences between lengths 6 and 12.

Q. Are any of your sequences unique i.e. not found anywhere in nature?

There are 20 natural amino-acids

```
aa <- c("A","R","N","D","C","E","Q","G","H","I",
        "L","K","M","F","P","S","T","W","Y","V")
```

```
generate_protein <- function(len) {

  #The amino-acids to sample from
  aa <- c("A","R","N","D","C","E","Q","G","H","I",
          "L","K","M","F","P","S","T","W","Y","V")

  #Draw n=len amino acids to make our sequence
  ans <- sample(aa, size=len, replace = T)
  ans <- paste(ans, collapse = "")
  return(ans)
}
```

```
myseq <- generate_protein(42)
myseq
```

```
[1] "WLTRRDMHMPADNQDGDHQQDVHRLGVQQTWQPSDKWCNSWNK"
```

Q. Use that function to generate some different sequences between lengths 6 and 12.

```
generate_protein(6)
```

```
[1] "KSMMQE"
```

```
generate_protein(7)
```

```
[1] "LQLPLDT"
```

```
generate_protein(8)
```

```
[1] "YASFNDQW"
```

```
generate_protein(10)
```

```
[1] "QARSLYEIGQ"
```

```
generate_protein(11)
```

```
[1] "TDYPGANFHRE"
```

```
generate_protein(12)
```

```
[1] "QQTLGEFPDRPA"
```

```
for(i in 6:12) {  
  #FASTA ID line ">ID"  
  cat(">", i, sep = "", "\n")  
  #Protein sequence line  
  cat(generate_protein(i), "\n")  
}
```

>6
KLQKTH
>7
GGDLAFR
>8
PNEWHQIV
>9
NEMASSSNE
>10
RLDAHKWDFR
>11
RPHLCSMARVS
>12
FHCYIDRFGNMK

Q. Are any of your sequences unique i.e. not found anywhere in nature?

A Majority of my sequences were unique, and not found in nature. Only the amino-acid sequences with a length of 6 and 7 amino-acids had 100% identity when searched through blastp.