Class 06: R Functions

Diego Diaz, PID: A17328629

Background

All functions in R have at least 3 things:

- A **name** used to call a function.
- One or more input **arguments**.
- The **body**, the lines of code that do the work.

Our first function:

Let's right a silly little function called `add()` to add some numbers (the input arguments.)

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

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

We cnan also set an argument to a default value, seen in the comment above.

Now we can use this function.

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

```
[1] 101
```

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

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

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

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

```
[1] 200   2
```

Q. What if put three inputs into the function?

```
#add(c(100,1), 1, 1)
```

A second function.

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

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

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

```
[1] 7 1 4 9
```

Q. Generate 9 random numbers taken from the input vector <- 1:10.

```
sample(c(1:10), size=9)
```

```
[1] 9 4 1 6 8 3 7 5 2
```

Q. Generate 12 random numbers taken from the input vector <- 1:10

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

```
 [1]  7  3  1  6  4  6  4 10  7  5  7  2
```

Q. Write code for the`sample()` function to generate nucleotide sequences length = 6

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

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

Q. Write a first function `generate.dna()` that returns a user specified length DNA sequence.

```
generate.dna <- function(len=6) {
  sample(x=c("A", "T", "G", "C"), size=len, replace=T)
}
```

```
cat((generate.dna(len=20)), sep="")
```

```
CAGTTAGTTACGATATAGCT
```

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

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

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

Modify and improve our function to return it's generated sequence in a more standard format like "AGTATATACC" rather than the vector "A", "T", "T"

```
nuc <- c("A", "T", "G", "C")

generate.dna <- function(len = 6, fasta = TRUE) {
  ans <- sample(nuc, size = len, replace = TRUE)

  if (fasta) {
    cat("Single-element vector output\n")
    ans <- paste(ans, collapse = "")
  } else {
    return(ans)
  }

  return(ans)
}
```

```
generate.dna(len=20)
```

```
Single-element vector output

[1] "GCGCACAGGCTCTACCCAGC"
```

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

```
paste(c("alice", "barry"), "loves R")
```

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

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

```
goodmood <- TRUE

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

```
Great!
```

Protein generating function

Q. Write a function that generates a user-specified length protein sequence.

Q. Use that function to generate random protein sequences between length 6 and 12.

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

```
#Amino acids for sampling.
aa <- c("A", "R", "N", "D", "C", "E", "Q", "G", "H", "I", "L", "K", "M", "F", "P", "S", "T",
#Draw from amino acids to make sequence.
generate.protein <- function(len=6) {
  ans <- sample(aa, size=len, replace=T)
  ans <- paste(ans, collapse="")
  return(ans)
}
```

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

```
>6
NQNMYK
>7
MGFLMNR
>8
ETNMKSHI
>9
SSLYCHQNP
>10
VGYCYPGHPY
>11
WFWYIHIGLRQ
>12
RFFHWFLFPDRL
```