R functions

Chris Yi (A16849780)

Today we will get more exposure to functions in R. We call functions to do all our work and today we will learn how to write our own.

First function

When setting default values to arguments, we don't have to supply them when we call our function.

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

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

```
[1] 2
```

```
add(1,c(10,100))
```

```
[1]  11 101
```

```
add0 <- function(x, y=0) {x+y}
```

```
add0(100)
```

```
[1] 100
```

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

```
addz(1, 2, 3)
```

```
[1] 6
```

Second function

Let's write a function that generates random nucleotide sequences.

We can make use of the in-built `sample()` function in R to help us here.

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

```
[1] 10  6  9  5  1  8  2  7  3
```

```
sample(1:10, size=11, replace = TRUE)
```

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

> Can you use `sample()` to generate a random nucleotide sequence of length 5.

```
sample(x=c("A","T","C","G"), size=5, replace = TRUE)
```

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

> Q. Write a function `generate_dna()` that makes a nucleotide sequence of a user specified length.

Every function in R has at least 3 things:

- a **name** (in our case, "generate_dna")
- one or more **input arguments** (the length of sequence we want)
- a **body** (R code that does the work)

```
generate_dna <- function(length=5) {
  bases <- c("A", "C", "T", "G")
  sample(bases, size=length, replace = TRUE)

}
```

```
generate_dna(100)
```

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

Q. Can you write a `generate_protein()` function that returns amino acid sequence of a user requested length?

```
aa <- bio3d::aa.table$aa1[1:20]
```

```
generate_protein <- function(length=20) {
  amino_acids <- aa
  sample(amino_acids, size=length, replace = TRUE)
}
```

```
generate_protein(100)
```

```
  [1] "D" "G" "R" "W" "W" "A" "A" "Q" "A" "M" "R" "V" "N" "Y" "V" "Q" "I" "A"
 [19] "R" "D" "R" "P" "E" "P" "Y" "Q" "N" "E" "W" "L" "Y" "I" "H" "T" "G" "P"
 [37] "K" "R" "A" "D" "W" "C" "L" "M" "Q" "R" "N" "Q" "C" "F" "D" "N" "F" "K"
 [55] "T" "N" "R" "I" "M" "R" "A" "G" "I" "V" "T" "F" "K" "E" "H" "Y" "H" "V"
 [73] "M" "I" "D" "C" "R" "H" "Q" "M" "G" "E" "S" "W" "P" "K" "T" "D" "D" "M"
 [91] "R" "F" "T" "S" "N" "F" "Q" "M" "F" "A"
```

I want my output of this function not to be a vector with one amino acid per element but rather a one element single string.

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

```
[1] "AGCT"
```

```
generate_protein <- function(length=20) {
  amino_acids <- aa
  s <- sample(amino_acids, size=length, replace = TRUE)
  paste(s, collapse="")
}
```

```
generate_protein()
```

```
[1] "HCYDQPGQGLSKCIDGKIIN"
```

Q. Generate protein sequences from length 6 to 12

We can use the useful utility function `sapply()` to help us "apply" our function over all the values 6 to 12.

```
ans <- sapply(6:12, generate_protein)
ans
```

```
[1] "FTDRRR"       "NEEACIQ"      "MRQNSISE"     "SYCMRHEDF"    "LWGDQGTKGR"
[6] "RFELKPKAENT"  "AYHMSLPFMTSC"
```

```
cat( paste(">ID.", 6:12, sep="","\n", ans, "\n") )
```

```
>ID.6
FTDRRR
 >ID.7
NEEACIQ
 >ID.8
MRQNSISE
 >ID.9
SYCMRHEDF
 >ID.10
LWGDQGTKGR
 >ID.11
RFELKPKAENT
 >ID.12
AYHMSLPFMTSC
```

Q. Are any of these sequences unique in nature - i.e. never found in nature. We can search "refseq-protein" and look for 100% Ide and 100% coverages matches wtih BLASTp.

Yes