Class 6: R Functions

AUTHOR
Christopher Leone (A16731724)

Introduction to Functions

Let's start writing our first function to add some numbers.

> Every R function has 3 important components:

- 1. the name, we get to choose.
- 2. input arguments (there could be any amount)
- 3. function body (where the code is).

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

I can now just use this function, as shown here:

```
x <- c(1:10)
y <- c(11:20)
add(x, y)
```

```
 [1] 12 14 16 18 20 22 24 26 28 30
```

Functions can have "required" input arguments and "optional" input arguments. Optional arguments are defined with an "equals default value" (`y=10`) in the function declination.

Writing a function to return a DNA sequence of a user specified length.

1. We can use the `sample()` function for ideas.

```
# genDNA <- function(size=5){}
students <- c("Jeff", "Jeremy", "Peter")
sample(students, size=5, replace = TRUE)
```

```
[1] "Jeff"   "Jeremy" "Jeremy" "Peter"  "Peter"
```

2. Let's try using this function to create nucleotide sequences.

```
# Must have "replace = TRUE" for repeats!
bases <- c("A", "T", "C", "G")
```

```
sample(bases, size=10, replace = TRUE)
```

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

3. Now we have a working snippet, so let's use it to generate our `function()`.

```
genDNA <- function(size=10){
  bases <- c("A", "T", "C", "G")
sample(bases, size, replace = TRUE)
}
genDNA(50)
```

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

4. Finally, to polish the output so it returns a sequence such as "ATGCATA..." if we want, otherwise return the raw output. Here, I want the collapsed sequence:

```
genDNA <- function(size=10, collapsed = FALSE){
  bases <- c("A", "T", "C", "G")
  seq <- sample(bases, size, replace = TRUE)
  if(!collapsed) {
    paste(seq)
  } else {
    paste(seq, collapse="")
  }
}
genDNA(50, TRUE)
```

```
[1] "AGGGACGATCTCGTCCGGACACTGCTTCAACTCACGTCAGTCAGAATGAA"
```

Writing a function to return protein sequences of a user specified length.

Similar to `genDNA()`, let's start using the sample function and build up from there. Main difference includes bringing the amino acids into play rather than 4 nucleotides.

We can get the set of 20 natural amino acids from the **bio3d** package.

```
head(bio3d::aa.table, 5)
```

```
    aa3 aa1    mass      formula          name
ALA ALA   A  71.078   C3 H5 N O1       Alanine
ARG ARG   R 157.194 C6 H13 N4 O1      Arginine
ASN ASN   N 114.103  C4 H6 N2 O2    Asparagine
ASP ASP   D 114.079   C4 H4 N O3 Aspartic Acid
CYS CYS   C 103.143 C3 H5 N O1 S       Cystein
```

Once called, let's use `aa.table` as our database for amino acids, and complete the function `genProt()`.

```
genProt <- function(size=6, collapsed=TRUE){
  amino <- bio3d::aa.table$aa1[1:20]
  seq2 <- sample(amino, size, replace = TRUE)
  if(collapsed) {
    paste(seq2, collapse="")
  } else {
    paste(seq2)
  }
}
genProt(size=25, TRUE)
```

```
[1] "YVEVLMYIADTNRRVYMWHHFPFNL"
```

Now, what if I wanted to generate a sequence of random length 6-12?

```
genProt2 <- function(collapsed=TRUE){
  # to set a random size of protein each time of length 6-12.
  randomSize <- sample(c(6:12), 1)

  # our protein parameters
  amino <- bio3d::aa.table$aa1[1:20]
  seq3 <- sample(amino, size=randomSize, replace = TRUE)

  # do we want a collapsed AA sequence?
  if(collapsed) {
    paste0(seq3, collapse="")
  } else {
    paste(seq3)
  }
}
genProt2(T)
```

```
[1] "EGIVRGR"
```

And if I wanted to print protein sequences of all lengths between 6-12?

```
prot <- sapply(6:12,FUN=genProt)
prot
```

```
[1] "GYINQR"       "SFELNTW"      "KFEESCTC"     "LIPSDYLDS"    "AEFIEWNIPL"
[6] "KSGFAWWDLFR"  "PVELTRWPLYHA"
```

It would also be cool and useful if I could get these in FASTA format for easy searching. Let's combine the functions of both the `cat()` and `paste()` functions.

```
fasta <- paste(">ID.", 6:12, "\n", prot, sep="")
cat(fasta, sep="\n")
```

```
>ID.6
GYINQR
>ID.7
SFELNTW
>ID.8
KFEESCTC
>ID.9
LIPSDYLDS
>ID.10
AEFIEWNIPL
>ID.11
KSGFAWWDLFR
>ID.12
PVELTRWPLYHA
```

Through a BLASTp query, I found that proteins 6 & 7 are NOT unique and can be found matched in the database (100% coverage and identity), but proteins 8-12 are in fact unique!