class 06

abe A16735986

Table of contents

1. Function Basics

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

Every R function has 3 things:

-name -input arguments (there can be loads of these separated by a comma) -the body (the R code that does the work)

```r
add <- function(x, y=10, z=0){
  x + y + z
}
```

I can just use this function like any other function as long as E knows about it (i.e. run the code chunk)

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

```
[1] 101
```

```r
add( x=c(1,2,3,4), y=100)
```

```
[1] 101 102 103 104
```

```r
add(1)
```

```
[1] 11
```

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

```r
add(x=1, y=100, z=10)
```

```
[1] 111
```

> Q. Write a function to return a DNA sequence of a user specified length? Call it `generate_dna()`

The `sample()` function can help here

```r
#generate_dna <- function(size=5){}

students <- c("jeff", "jeremy", "peter")

sample(students, size = 5, replace=TRUE)
```

```
[1] "jeff"   "peter"  "peter"  "jeremy" "jeremy"
```

2. Generate DNA Sequences

Now work with `bases` rather than `students`

```r
bases <- c("A", "C", "G", "T")
sample(bases, size=25, replace=TRUE)
```

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

Now I have a working 'snippet' of code that I ran to the body of the first function version here:

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

```r
generate_dna(100)
```

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

```r
generate_dna()
```

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

I want the ability to return a sequence like "AGTACCTG" I.E. A ONE ELEMENT VECTOR where the bases are all together

```r
generate_dna <- function(size=5, together=TRUE){
  bases <- c("A", "C", "G", "T")
  sequence <- sample(bases, size=size, replace=TRUE)
  if(together){
 sequence <- paste(sequence, collapse = "")
  }
  return(sequence)
}
```

```r
generate_dna(together = FALSE)
```

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

3. Generate Protein Function

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

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

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

> Q. Write a protein sequence generating function that will return sequences of a user specified length?

```r
generate_protein <- function(size=10, together=TRUE){
  aa <- c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I", "L", "K", "M", "F", "P", "S", "T
  sequence <- sample(aa, size=size, replace=TRUE)
  if(together){
 sequence <- paste(sequence, collapse = "")
  }
  return(sequence)
}
```

```r
generate_protein()
```

```
[1] "QNRPFGFMDS"
```

> Q. Generate random protein sequences of length 6 to 12 amino acids.

```r
generate_protein(8)
```

```
[1] "RYLLNVWF"
```

We can fix this inability to generate multiple sequences by either editing and adding to the function body code (i.e. a for loop) or by using the R **apply** family of utility functions.

```r
sapply(6:12, generate_protein)
```

```
[1] "CCWMGA"       "WYAVWSH"      "CQWYIPAS"     "HEEFREAWK"    "CMAYFGYECQ"
[6] "IHCGQGYPYPW"  "ADLKPWTPYEVM"
```

It would cool and useful if I could get FASTA format output

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

```
[1] "TPDDCN"       "QYNKWYV"      "YKHMQYLC"     "MTVPCRFNM"    "KKKHMAIEFA"
[6] "LFRVYRPWPKQ"  "WRGGALSWESGW"
```

```r
cat(ans, sep="\n")
```

```
TPDDCN
QYNKWYV
YKHMQYLC
MTVPCRFNM
KKKHMAIEFA
LFRVYRPWPKQ
WRGGALSWESGW
```

I want this to look like

```
>ID.6
YQKPFD
ID.7
QGWWMCA
LLNSWHLN
ID.8
QLYTWVLVI
etc
```

The functions `paste()` and `cat()` may be helpful

```r
id.line <- paste(">ID.", 6:12, sep="")
id.line
```

```
[1] ">ID.6"  ">ID.7"  ">ID.8"  ">ID.9"  ">ID.10" ">ID.11" ">ID.12"
```

```r
id.line <- paste(">ID.", 6:12, sep="")
seq.line <- paste(id.line, ans, sep="\n")
cat(seq.line, sep="\n")
```

```
>ID.6
TPDDCN
>ID.7
QYNKWYV
>ID.8
YKHMQYLC
>ID.9
MTVPCRFNM
>ID.10
KKKHMAIEFA
>ID.11
LFRVYRPWPKQ
>ID.12
WRGGALSWESGW
```

> Q. Determine if these sequences can be found in nature or are the unique? Why or why not?

I BLASTp searched my FASTA format sequences against NR and found that length 6, 7, and 8 are not unique due to them having a quary cover and percent identity of 100%. However, for lengths 9, 10, 11, and 12, they are unique, as they do not have a protein that has both query cover of 100% and percent identity of 100%.