Class06

Anyoleth Alarcon (A17347293)

Table of contents

1. Function Basics

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

Every R function has 3 components:

- name (we get to pick this)
- input arguments (there can be loads of these separated by comma)
- the body (the R code that does the work)

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

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

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

```
[1] 101
```

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

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

```
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.

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

```
[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

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

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

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

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

Now work with `bases` rather than `students`

```
bases <- c("A", "C", "G", "T")
sample(bases, size=10, replace=TRUE)
```

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

2. Generate DNA Sequence

Now I have a working `snippet` of code I can use this as the body of my first function version here:

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

```
generate_DNA(100)
```

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

```
generate_DNA(100)
```

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

I want the ability to return a sequence like "AGTACCTG" i.e. a one element vector where the bases are all together.

```
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)
}
```

```
generate_DNA(together=FALSE)
```

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

```
generate_DNA(20)
```

```
[1] "TATAAGAGTAAATGCCGCCT"
```

3. Generate Protein Function

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

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

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

```
generate_protein <- function(size=6, together=TRUE){

  ##Get the 20 amino-acids as a vector
  aa <- bio3d::aa.table$aa1[1:20]
  sequence <- sample(aa, size=size, replace=TRUE)

  ##Optionally return a single element string
  if(together){
  sequence <- paste(sequence, collapse = "")

  }
  return(sequence)
}
```

```
generate_protein(together = F)
```

```
[1] "C" "R" "E" "F" "C" "Y"
```

```
generate_protein(7)
```

```
[1] "PRVETML"
```

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

```
##Generate_protein(size=6:12)
```

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

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

```
[1] "KLAVIP"       "VDDRLNW"      "LTTGTPHF"     "AGCEHNGTL"    "CQLCTNVIQC"
[6] "FMHPEYAFDES"  "EHDWGFQDLFSI"
```

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

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

```
[1] "RYGQYA"       "LDWFHVS"      "QVVWKQLY"     "GETPECPMS"    "FHSCWQLPHT"
[6] "YMQFEPPPKHT"  "WSTYFNFTDQQW"
```

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

```
RYGQYA
LDWFHVS
QVVWKQLY
GETPECPMS
FHSCWQLPHT
YMQFEPPPKHT
WSTYFNFTDQQW
```

I want this to look like

```
>ID.6
PADREN
>ID.7
RTNVGPT
>ID.8
YNGFMNYF
MECRCGCCW
VCDDMGTDHN
WQDKVHTAYDA
WSHAQVWWKGLT
and so forth
```

The functions `paste()` and `cat()` can help us here...

```
paste(">ID.", 7:12, sep = "")
```

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

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

```
>ID.6
RYGQYA
>ID.7
LDWFHVS
>ID.8
QVVWKQLY
>ID.9
GETPECPMS
>ID.10
FHSCWQLPHT
>ID.11
YMQFEPPPKHT
>ID.12
WSTYFNFTDQQW
```

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

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

```
seq.line <- paste(id.line, ans, sep="\n")
cat(seq.line, sep="\n")
```

```
>ID.6
RYGQYA
>ID.7
LDWFHVS
>ID.8
QVVWKQLY
>ID.9
GETPECPMS
>ID.10
FHSCWQLPHT
>ID.11
YMQFEPPPKHT
>ID.12
WSTYFNFTDQQW
```

Q. Determine if these sequences can be found in nature or are they unique?

I BLASTp searched my FASTA format sequences against NR and found that length 6 and 7 are not unique and can be found in the databases with 100% coverage and 100% identity.

Random sequence of length of 8 and above are unique and cannot be found in the databases.