

Class 6: R functions

Yoonjin Lim (PID: A16850635)

Table of contents

1. Function basics	1
2. Generate DNA sequence	2
3. Generate Protein function	3

1. Function basics

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

Every R function has 3 things:

- name (we get to pick this)
- input arguments (there can be loads of these separated by a 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(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

```
#generate_dna <- function(size=5){ }  
  
students <- c("jeff", "jeremy", "peter")  
  
sample(students, size = 5, replace=TRUE)
```

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

2. Generate DNA sequence

Now work with bases rather than students

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

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

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

```
generate_dna()
```

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

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()
```

```
[1] "GTGTT"
```

```
generate_dna(together=F)
```

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

3. Generate Protein function

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

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

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

```
generate_protein <- function(size=6, together=TRUE){  
  
  ## Get the 20 amino-acids as a vector  
  aa <- bio3d::aa.table$aa1[1:20]  
  sequence_protein <- sample(aa, size=size, replace=TRUE)  
  
  ## Optionally return a single element string  
  if(together) {  
    sequence_protein <- paste(sequence_protein, collapse= "")  
  }  
  return(sequence_protein)  
}
```

```
generate_protein()
```

```
[1] "PFRCCCT"
```

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

```
generate_protein(7)
```

```
[1] "CTDLVIC"
```

```
generate_protein(8)
```

```
[1] "QLPVGTEH"
```

```
generate_protein(9)
```

```
[1] "TDTIWDPTI"
```

We can fix this inability to generate multiple sequences by either editing and adding to 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] "EPLRCA"      "FLPTPKF"      "MRTATDAF"      "LKPIFRFVE"      "TTVLGHIKEE"  
[6] "MEDCFWAIGYC" "WRFHPCMFETNW"
```

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

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

```
[1] "GFHWHH"      "ITFTVLP"      "NAGWMPYP"      "RFPFLSRHA"      "CDFVKHFWDF"  
[6] "LEESHAVKVVA" "CHQNYQYNIGMT"
```

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

```
GFHWHH  
ITFTVLP  
NAGWMPYP  
RFPFLSRHA  
CDFVKHFWDF  
LEESHAVKVVA  
CHQNYQYNIGMT
```

I want this to look like

```
>ID.6  
YEMMAH  
>ID.7  
FCMSIMN  
>ID.8  
QLDAFIGY
```

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

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

```

>ID.6
GFHWHH
>ID.7
ITFTVLP
>ID.8
NAGWMPYP
>ID.9
RFPFLSRHA
>ID.10
CDFVKHFWDF
>ID.11
LEESHAVKVVA
>ID.12
CHQNYQYNIGMT

```

```

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

```

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

```

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

```

```

>ID.6
GFHWHH
>ID.7
ITFTVLP
>ID.8
NAGWMPYP
>ID.9
RFPFLSRHA
>ID.10
CDFVKHFWDF
>ID.11
LEESHAVKVVA
>ID.12
CHQNYQYNIGMT

```

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

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

I BLASTp searched my FASTA format sequences against NR and found that length 6, 7, 8 are not unique and can be found in the databases with 100% coverage and 100% identity.

Random sequences of length 9 and above are unique and can't be found in the databases.