Lab06

Karina Cardenas, A16742606

Table of contents

Intro to Functions

Basics of Functions

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 many of them separated by a comma)
- the body (The R code that does the work)

basic function

```
#setting a default for y
add <- function(x, y = 100, z = 10){
  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] 111
```

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

```
[1] 111 112 113 114
```

```
add(1)
```

```
[1] 111
```

Functions can have "required" input arguments and "optional" input arguments. In this case x is "required" to execute and y is "optional" as a default has been assigned as a fallback for a missing y input.

Note: The optional arguments are defined with an equal default value(y = 10)

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

```
[1] 111
```

This code **will not execute** and throw out an error as the original function does not have a third argument of z.

Writing functions

Function 1- DNA

Q. Write a function to return a DNA sequence of user specified length?

The `sample()` function can help here

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

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

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

Q. Now work with `bases` rather than `students`

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

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

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

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

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

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

```
generate_dna(100)
```

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

Function 2 - grouping

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

```
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] "TTTAA"
```

```
generate_dna(together = TRUE)
```

```
[1] "TAAAA"
```

Function 3 - protein

Q. Create a `generate_protein()` function

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

```
bio3d::aa.table
```

```
    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
GLN GLN   Q 117.126   C4 H9 N2 O2                     Glutamine
GLU GLU   E 128.106    C5 H6 N O3                 Glutamic Acid
GLY GLY   G  57.051    C2 H3 N O1                       Glycine
HIS HIS   H 137.139   C6 H7 N3 O1                     Histidine
ILE ILE   I 113.158   C6 H11 N O1                    Isoleucine
LEU LEU   L 113.158   C6 H11 N O1                       Leucine
LYS LYS   K 129.180  C6 H13 N2 O1                        Lysine
MET MET   M 131.196  C5 H9 N O1 S                    Methionine
PHE PHE   F 147.174    C9 H9 N O1                 Phenylalanine
PRO PRO   P  97.115    C5 H7 N O1                       Proline
SER SER   S  87.077    C3 H5 N O2                        Serine
THR THR   T 101.104    C4 H7 N O2                     Threonine
TRP TRP   W 186.210 C11 H10 N2 O1                     Tryptophan
TYR TYR   Y 163.173    C9 H9 N O2                      Tyrosine
VAL VAL   V  99.131    C5 H9 N O1                        Valine
ABA ABA   X  85.104   C4 H7 N1 O1 alpha-aminobutyric acid
ASH ASH   D 115.087    C4 H5 N O3   Aspartic acid Neutral
```

```
CIR CIR   R 157.170  C6 H11 N3 O2                      citrulline
CME CME   C 179.260 C5 H9 N O2 S2 s,s-(2-hydroxyethyl)thiocysteine
CMT CMT   C 115.154  C4 H5 N O1 S                  o-methylcysteine
CSD CSD   C 134.134  C3 H4 N O3 S           s-cysteinesulfinic acid
CSO CSO   C 119.142  C3 H5 N O2 S                 s-hydroxycysteine
CSW CSW   C 135.142  C3 H5 N O3 S                cysteine-s-dioxide
CSX CSX   C 119.142  C3 H5 N O2 S                    s-oxy cysteine
CYM CYM   C 102.135  C3 H4 N O1 S                  Cystein Negative
CYX CYX   C 102.135  C3 H4 N O1 S                    Cystein SSbond
DDE DDE   H 280.346 C13 H22 N5 O2                       diphthamide
GLH GLH   E 129.114    C5 H7 N O3            Glutatmic acid Neutral
HID HID   H 137.139   C6 H7 N3 O1                          Histidine
HIE HIE   H 137.139   C6 H7 N3 O1                          Histidine
HIP HIP   H 138.147   C6 H8 N3 O1                Histidine Positive
HSD HSD   H 137.139   C6 H7 N3 O1                          Histidine
HSE HSE   H 137.139   C6 H7 N3 O1                          Histidine
HSP HSP   H 138.147   C6 H8 N3 O1                Histidine Positive
IAS IAS   D 115.087    C4 H5 N O3                     beta-aspartyl
KCX KCX   K 172.182  C7 H12 N2 O3         lysine nz-carboxylic acid
LYN LYN   K 129.180  C6 H13 N2 O1                    Lysine Neutral
MHO MHO   M 147.195  C5 H9 N O2 S                  s-oxymethionine
MLY MLY   K 156.225  C8 H16 N2 O1                 n-dimethyl-lysine
MSE MSE   M 178.091 C5 H9 N O1 SE                  selenomethionine
OCS OCS   C 151.141  C3 H5 N O4 S             cysteinesulfonic acid
PFF PFF   F 165.164  C9 H8 F N O1          4-fluoro-l-phenylalanine
PTR PTR   Y 243.153 C9 H10 N O5 P                 o-phosphotyrosine
SEP SEP   S 167.057  C3 H6 N O5 P                     phosphoserine
TPO TPO   T 181.084  C4 H8 N O5 P                  phosphothreonine
```

```
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) {

  #Gets 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 of string
  if(together){
```

```
    sequence <- paste(sequence, collapse = "")
  }

  return(sequence)
}
```

```
generate_protein(15)
```

```
[1] "CDIMTMAFCACEGHE"
```

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

We can fix this inability of looping through the function by using `sapply()`. This removes the need to hard edit/code.

```
# X = lengths 6 - 12, FUN = generation_protein
sapply(6:12, generate_protein)
```

```
[1] "ESYNHN"       "FPIIYDP"      "CRDNKASM"     "LTKRVKGGD"    "RQERYLYHFD"
[6] "NNGSPGKPHPM"  "CHCVYTTKLTWK"
```

Function 4 - FASTA format

It would be useful if I could get FASTA format output. I want this to look like

```
>ID.6
DVIIYG
>ID.X
XXXXXX
```

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

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

```
HNTFET
TYMMVPH
DQCPPNQT
TGHGTAYMH
LTVKPEYFEE
QRDCQSACRPS
CRHLRNAGMLIS
```

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

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

```
>ID.6
HNTFET
>ID.7
TYMMVPH
>ID.8
DQCPPNQT
>ID.9
TGHGTAYMH
>ID.10
LTVKPEYFEE
>ID.11
QRDCQSACRPS
>ID.12
CRHLRNAGMLIS
```

A more simpler approach...

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

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

cat(seq.line, sep = "\n")
```

```
> ID.6
HNTFET
> ID.7
TYMMVPH
> ID.8
DQCPPNQT
> ID.9
TGHGTAYMH
> ID.10
LTVKPEYFEE
> ID.11
QRDCQSACRPS
> ID.12
CRHLRNAGMLIS
```

Q. Determine if these sequences can be found

I Blastp searched my FASTA format sequences against NR and found that lengths 6,7 and 8 are unique with a 100% coverage and 100% identity. On the other hand, 9 - 12 lengths fail to have 100% coverage showing they do not exist in nature.

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