Class 6: R functions

Emily Chen (PID: A16925878)

Table of contents

1. Function Basics

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

Every R function has 3 things:

- name (we get to pick this)
- input argument (there can be lots of these separated by a comma)
- the body (the R code that deos the work)

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

I can just use the function like any other functions as loung as R knows about it (i.e make sure to run the code chunk first)

```
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 equal default value (`y=10`) in the function definition. The codes with the equal sign are not required for the functions

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

```
[1] 111
```

> Write a function to retuns a DNA sequnce of a user specified length? Call it `generate_dna`

We can't take a sample larger than the population but we can do so when we add `replace=TRUE`

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

student <- c("jeff", "jermey", "peter")

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

```
[1] "jermey" "peter"  "jeff"   "jermey" "peter"
```

Now work with `bases` rather than `students`

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

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

Now I have a working `snippet` of cide I can use this as the body of my first functions wersion here:

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

```
generate_dna ()
```

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

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] "CGAAC"
```

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

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

3. Generate Protein Functions

We can get the set of 20 natural amino acids from the **bio3d** package. To install use the code `install.packages("bio3d")`

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

> Q. Write a protein sequence generating functions that will return sequence 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 <- sample(aa, size, replace=TRUE)

  ## optionally return a single element string
  if(together) {
    sequence<- paste(sequence, collapse= "")
  }
  ## do not put () after sequence or else it will be made into a function and that is not wh
  return(sequence)
}
```

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

```
generate_protein(6)
```

```
[1] "DEENEN"
```

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

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

```
[1] "TCIYFQ"       "FIQDGYF"      "FLWNDSES"     "QARKTHEMW"    "FFILHNPEQR"
[6] "YRIIDFWMANP"  "KKWWCNVHIGVD"
```

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

```
TCIYFQ
FIQDGYF
FLWNDSES
QARKTHEMW
FFILHNPEQR
YRIIDFWMANP
KKWWCNVHIGVD
```

cat() function will mke you the list and then`sep()` in a function that will tell R how to format the list, such as we have sep=“ “, it will separate the output with” ” ans is the protein sequence

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

```
[1] ">ID.7TCIYFQ"        ">ID.8FIQDGYF"       ">ID.9FLWNDSES"
[4] ">ID.10QARKTHEMW"    ">ID.11FFILHNPEQR"   ">ID.12YRIIDFWMANP"
[7] ">ID.7KKWWCNVHIGVD"
```

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

I want it to look like this

```
>ID.6
HLDWLV
>ID.7
VREAIQN
>ID.8
WPRSKACN
```

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

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

I BLASTp searched my FASTA format sequence against NR and found that lengths 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. Lots of the binding grooves, such as for MHC, are 9 amino acids long; thus, they are unique.