Class 6: R Functions

Michael McClellan (PID: 16962395)

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 commas)
- 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 R 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 us here

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

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

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

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

2. Generate DNA sequence

Now work with `bases` rather than `students`

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

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

Now I have a working 'snippet' of code I can use this as the bady of my 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()
```

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

I want the ability to return a squence 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()
```

```
[1] "AGCTG"
```

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

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

3. Generate Protein function

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

```r
library(bio3d)
aa <- bio3d::aa.table$aa1[1:20]
```

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

```r
generate_protein_sequence <- function(size=5, together=TRUE) {
  ## Get the 20 amino-acids as a vector
  aa <- bio3d::aa.table$aa1[1:20]
  protein_sequence <- sample(aa, size=size, replace=TRUE)

  if(together) {
  protein_sequence <- paste(protein_sequence, collapse = "")
  }
  return(protein_sequence)
}
```

```r
generate_protein_sequence()
```

```
[1] "WWRNT"
```

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

```r
generate_protein_sequence(7)
```

```
[1] "YCWFSRC"
```

```r
# generate_protein_sequence(size=6:12)
```

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.

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

```
[1] "LILMGN"       "WSKIPWC"      "QSNMTACT"     "NIKKEGRRA"    "DQIHTTGADE"
[6] "CKTTFTIITTH"  "ENRVRREEYICW"
```

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

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

```
[1] "YPFRNW"       "SPQYIHQ"      "DRQPEIMP"     "LTHDRTSQQ"    "HHLDDCNRWD"
[6] "QQWLSWQNQAF"  "IEPGGALDGHLA"
```

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

```
YPFRNW
SPQYIHQ
DRQPEIMP
LTHDRTSQQ
HHLDDCNRWD
QQWLSWQNQAF
IEPGGALDGHLA
```

I want this to look like:

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

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

```r
paste(">ID.", 6:12, ans, sep="")
```

```
[1] ">ID.6YPFRNW"       ">ID.7SPQYIHQ"      ">ID.8DRQPEIMP"
[4] ">ID.9LTHDRTSQQ"    ">ID.10HHLDDCNRWD"  ">ID.11QQWLSWQNQAF"
[7] ">ID.12IEPGGALDGHLA"
```

```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", 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 refseq_protein and found that length 6 and 7 are not unique and can be found in the databases with 100% query coverage and 100% identity.

Random sequences of length 8 and above are unique and can't be found in the databases.