Class 6: R functions

Barry (PID: 911)

Table of contents

1. Function basics

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

Every R function has 3 things:

- name (we get to pick this)
- input arguments (there can loads of these sperated 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 defination.

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

```
[1] 111
```

> Q. Write a function to return a DNA sequence of a user specified lenght? 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"   "peter"  "jeremy" "jeff"
```

2. Generate DNA sequence

Now work with `bases` rather than `students`

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

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

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

```
generate_dna()
```

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

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

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

3. Generate Protein function

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

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

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

```
 [1] "A" "R" "N" "D" "C" "Q" "E" "G" "H" "I" "L" "K" "M" "F" "P" "S" "T" "W" "Y"
[20] "V"
```

and use this in our function

```
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 singe element string
  if(together){
    sequence <- paste(sequence, collapse = "")
  }
  return(sequence)
}
```

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

```
[1] "N" "G" "Y" "I" "C" "P"
```

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

```
generate_protein(7)
```

```
[1] "SPMSLPK"
```

```
generate_protein(8)
```

```
[1] "IDLCGMIQ"
```

```
generate_protein(9)
```

```
[1] "ADDDYRDIN"
```

```
# generate_protein(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.

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

```
[1] "KPWGQN"       "TMVWGYH"      "LLTIVPGG"     "MDYCKDEIQ"    "EHFDELGAKQ"
[6] "TSMVHRTKQCK"  "VDIDPENGSQEM"
```

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

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

```
[1] "LVMFVP"       "CQNNGFW"      "ERSCMPEV"     "PRFIHGNMW"    "DFWFMYNDVI"
[6] "EKMSVVFGINP"  "LDNIRNVCTVST"
```

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

```
LVMFVP
CQNNGFW
ERSCMPEV
PRFIHGNMW
DFWFMYNDVI
EKMSVVFGINP
LDNIRNVCTVST
```

I want this to look like FASTA format with an ID line, e.g.

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

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

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

```
>ID.7
LVMFVP
>ID.8
CQNNGFW
>ID.9
ERSCMPEV
>ID.10
PRFIHGNMW
>ID.11
DFWFMYNDVI
>ID.12
EKMSVVFGINP
>ID.7
LDNIRNVCTVST
```

```
id.line <- paste(">ID.",7:12, sep="")
id.line
```

```
[1] ">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", 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 foramt sequencs 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.