Class 6: R functions

Christopher Levinger (A17390693)

toc: true

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(1,100,10)
```

```
[1] 111
```

2 Generate DNA function

> Q: Write a function to return a nucleotide sequence of a user specified length? Call it `generate_dna()` The " function can help here

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

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

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

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

Now work with `bases` rather than `students`

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

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

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

```
generate_dna()
```

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

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) {
  paste(sequence, collapse="")
}
return(sequence)
}
```

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

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

3. Generate Protein function

> Q. Write a protein sequence generating function that will return sequences of a user specified length?
>
> Q Generate random protein sequences of length 6 to 12 amino acids
>
> Q Determine if these sequences can be found in nature We can get the set of 20 natural amino-acids from the "bio3d" package

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

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

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

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

```
[1] "MGTKTQ"       "TFRLNKE"      "NQENGPDV"     "ECDWKNPDW"    "NLMPYSCSDV"
[6] "EKIEAWTEGVN"  "YNLDMLSMTMWM"
```

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

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

```
[1] "HTLMPF"       "EATKMRF"      "NWCQISLD"     "MFYTDTDTK"    "VLDGQTYRQD"
[6] "GMHVVGYSKCC"  "ANTCVPVVIPAK"
```

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

```
HTLMPF/nEATKMRF/nNWCQISLD/nMFYTDTDTK/nVLDGQTYRQD/nGMHVVGYSKCC/nANTCVPVVIPAK
```

I want this to look like FAFSTA format:

```
>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/nHTLMPF
>ID.8/nEATKMRF
>ID.9/nNWCQISLD
>ID.10/nMFYTDTDTK
>ID.11/nVLDGQTYRQD
>ID.12/nGMHVVGYSKCC
>ID.7/nANTCVPVVIPAK
```

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

```
paste(id.line, ans, sep="\n")
```

```
[1] ">ID.6\nHTLMPF"         ">ID.7\nEATKMRF"        ">ID.8\nNWCQISLD"
[4] ">ID.9\nMFYTDTDTK"      ">ID.10\nVLDGQTYRQD"    ">ID.11\nGMHVVGYSKCC"
[7] ">ID.12\nANTCVPVVIPAK"
```

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

```
>ID.6
HTLMPF
>ID.7
EATKMRF
>ID.8
NWCQISLD
>ID.9
MFYTDTDTK
>ID.10
VLDGQTYRQD
>ID.11
GMHVVGYSKCC
>ID.12
ANTCVPVVIPAK
```

> Q: Determine if thse sequences can be found in anture or are they unique? Why or why not? I BLASTp searched my FAFSTA format sequences against NR and found that length 6,7,8, are not unique and can be found in databases with 100% coverage and 100% identity.

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