class 6: R functions

Bryn(PIDA69038039)

Today we are going to get more exposure to functions in R.

Lets start with a silly simple function to add some numbers:

```
add <- function(x, y){
 x+y
}
```

Can we use this function?

```
add(1,1)
```

```
[1] 2
```

```
add(c(100,200),1)
```

```
[1] 101 201
```

You can set a default number if you have a constant number you want to use in your function. For example:

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

```
log(10)
```

```
[1] 2.302585
```

```
log(10, base=10 )
```

```
[1] 1
```

```
add(100,1)
```

```
[1] 101
```

Lets have a look at teh `sample` function..

Question. What does it do?

the `sample` function in R randomly selects elements from a vector. It has two main uses:

```
sample(1:10, size=5)
```

```
[1] 10  6  8  7  9
```

What if I want 11 things drawn from my vectow 1 to 10

It wont work, because there are only ten options, so you cannot choose more than the size of the vector.

unless... You argue with the default that replace=TRUE instead of the default replace=FALSE

```
sample(1:10, size=11, replace=T)
```

```
 [1]  1  8  9 10  3  4  9  9  1  7  1
```

sidenote:

If you want to scan your sample in increments use seq function:

```
seq(5,50,by=3)
```

```
 [1]  5  8 11 14 17 20 23 26 29 32 35 38 41 44 47 50
```

Generate DNA sequences

> Q: Write a function to generate a random nucleotide sequence of a user specified size/length.

```
x <- c("A", "T","G","C")
sample(x, size=4)
```

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

what if you want to make a longer sequence?

```
sample(x, size=10, replace = T)
```

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

All functions in R have these three things: -a **name** (you pick this) -input **arguments** ("Length" of the output sequence) -function **body** (where the work gets done, line by line)

```
generate_DNA <- function(length){x <- c("A", "T","G","C")
sample(x, size=length,replace= T)}

generate_DNA(40)
```

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

I would like my function to print out something like: ATCGCGTA (single-element vector). To help with this I can use the `paste` function.

```
paste(generate_DNA(10), collapse = "")
```

```
[1] "ACTGCACTCC"
```

Now put it all together in one fucntion

```
generate_DNA <- function(length=10){
# the nucleotides to draw/sample from
x<- c("A", "T","G","C")
#Draw n=length nucleotides to make our sequence
ans <- sample(x, size=length,replace= T)
#Paste/join sequence into one word
ans <- paste(ans, collapse = "")
return(ans)
}

generate_DNA(length=9)
```

```
[1] "CTCGTCGCG"
```

```
generate_DNA(length=12)
```

```
[1] "TGCTCTGGCCCG"
```

I want the ability to switch between these two output formats. I can do this with an extra input argument to my function that controls this with TRUE/FALSE.

```
generate_DNA <- function(length=10, collapse=TRUE){
# if collapse is true it will collapse " " but if collapse is FALSE it will not.
# the nucleotides to draw/sample from
x<- c("A", "T","G","C")
#Draw n=length nucleotides to make our sequence
ans <- sample(x, size=length,replace= T)
#Paste/join sequence into one word
if(collapse) {
  ans <- paste(ans, collapse = "")
}
return(ans)
}

generate_DNA(10)
```

```
[1] "TCAGGAGATA"
```

> Q. Add the ability to print a wee message to add if the user is sad. Control this with a new input parameter called `mood`.

```
generate_DNA <- function(length=10, collapse=TRUE, mood=TRUE){
# if collapse is true it will collapse " " but if collapse is FALSE it will not.
# the nucleotides to draw/sample from
x<- c("A", "T","G","C")
#Draw n=length nucleotides to make our sequence
ans <- sample(x, size=length,replace= T)
#Paste/join sequence into one word
if(collapse) {
  ans <- paste(ans, collapse = "")
}
if(mood) {
  cat("Happy")
}
return(ans)
}

generate_DNA(10, mood=T)
```

```
Happy
```

```
[1] "AGTTTATTAC"
```

Q. Write a protein sequence generating function with the ability to output random amino acid sequences of a user definded length.

```
aa <- c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I",
                "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V")

length(aa)
```

```
[1] 20
```

```
generate_protein <- function(length=10, collapse=T){
  aa <-  c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I",
                "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V")
  prot <- sample(aa, size=length, replace=T)
if (collapse){
  prot <- paste(prot, collapse = "")
}
```

```
  return(prot)
}

generate_protein(6,T)
```

```
[1] "EQYAID"
```

Q. generate protein sequences form length 6-12 amino acids long.

```
generate_protein(12, T)
```

```
[1] "RDQMGHWMRMSW"
```

Generate_protein(6:12, T) doesnt work because my function is not vectorized (in other words, setup to work on each element of the first input argument `length`.

in particular, we can use `sapply`

sapply() in R is a vectorized function that applies a function to each element of a list or vector and attempts to simplify the result

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

```
[1] "EIGFWE"       "QGANPLI"      "ETHMVAQI"     "LSDHSATNE"    "WMNLQLPLRT"
[6] "EDCERHSTAYL"  "MRCCSEFICLEQ"
```

Q. Are any of these sequences unique in the sense that they have never been found in mature?

To make this accessible for blast lets get out sequences in FASTA format.

FASTA format looks like this: >id.6 JSDLIHDLIHE >id.7 KGALDIAHILI

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

```
[1] "ASAIYH"       "ANEFYRM"      "GWKNILCC"     "HQTYDGAHY"    "KHATNKIELG"
[6] "HWRRNNCTDRE"  "SMLWPKRWVKMY"
```

The functions `paste()` and `cat()` will help here

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

```
>id.6
ASAIYH
>id.7
ANEFYRM
>id.8
GWKNILCC
>id.9
HQTYDGAHY
>id.10
KHATNKIELG
>id.11
HWRRNNCTDRE
>id.12
SMLWPKRWVKMY
```

```
library(bio3d)

myseqs.vec <- sapply(6:12, generate_protein, collapse=T)
x <- as.matrix(myseqs.vec)
x
```

```
     [,1]
[1,] "VESQHT"
[2,] "FEQWSNE"
[3,] "NQGHGFNK"
[4,] "ACRLFLERH"
[5,] "NQPEEPTPFR"
[6,] "LARWDLLYPFT"
[7,] "PKSAPANGSFRD"
```

Yes, Sequence at length 9 starts to become unique.