Class06: R functions

Montserrat (PID: A16536527)

Table of contents

All functions in R have at least 3 things:

-A **name**, we pick this and use it to call our function -input **argument** (there can be multiple) - The **body** lines of R code that do the work

Our first (silly) function

write a function to add some numbers

```r
add <- function (x,y=1){
  x + y
}
```

Now we call this function

```r
add(10,100)
```

```
[1] 110
```

```r
add(c(10,10),100)
```

```
[1] 110 110
```

A second function

write a function to generate random nucleotide sequences of a user specified length:

The `sample` function can be helpful here

```r
sample(c("A","C","G","T"), size=50, replace = TRUE)
```

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

i want the a 1 element long character vector that looks "GCTATT" not "G" "C" "T" "A" "T" "T"

```r
v <- sample(c("A","C","G","T"), size=50, replace = TRUE)
paste(v,collapse="")
```

```
[1] "CGGATCGAAATTCGGTATACAGCGCGCCGCGTTGGACTTTTCTTGCACTT"
```

Turn this into my first wee function

```r
generate_dna <- function(size=50){
  v <- sample(c("A","C","G","T"), size=size, replace = TRUE)
paste(v,collapse="")
}
```

Test it:

```r
generate_dna(6)
```

```
[1] "ATTATT"
```

```r
generate_dna(60)
```

```
[1] "ATGCACCGAACATATGCGGAGAGAGATAAGGTTACTCTCGGCGCGCGGCGACACTCCAGT"
```

```r
if(TRUE){
  cat("HELLO You!")
}
```

```
HELLO You!
```

```r
fasta<- TRUE
if(fasta){
  cat("HELLO You!")
}
```

```
HELLO You!
```

Add the ability to return a multi-element vector or a single element fasta like vector

```r
generate_fasta <- function(size=50,fasta=TRUE){
  v <- sample(c("A","C","G","T"), size=size, replace = TRUE)
paste(v,collapse="")
if(fasta){
  cat("HELLO You!")
}
}
```

```r
generate_fasta(10)
```

```
HELLO You!
```

```r
generate_fasta(10,fasta=FALSE)
```

```r
generate_fasta <- function(size=50,fasta=TRUE){
  v <- sample(c("A","C","G","T"), size=size, replace = TRUE)
s<-paste(v,collapse="")
if(fasta){
  return(s)
}else{
  return(v)
}
}
```

```r
generate_fasta(10)
```

```
[1] "ACATTTTATC"
```

```r
generate_fasta(10,fasta=FALSE)
```

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

A protein generating function

```r
generate_protein <- function(size=50, fasta=TRUE){
  aa_codes <- c("A","R","N","D","C","Q","E","G","H","I","L","K","M","F","P","S","T","W","Y",
  v <- sample(aa_codes, size=size, replace=TRUE)
  s <- paste(v, collapse="")
  if(fasta){
    return(s)
  } else {
    return(v)
  }
}
```

```r
generate_protein(6)
```

```
[1] "YRLRHW"
```

generate all sequences between length of 6 and 12 using `generate_protein()`

one function to do this is "brute force"

```r
generate_protein(6)
```

```
[1] "PYSPSM"
```

```r
generate_protein(7)
```

```
[1] "DGEWYGH"
```

```
generate_protein(8)
```

```
[1] "NSSCRCRI"
```

A second way is to use a `for()` loop:

```
lengths <-6:12
for(i in lengths){
  cat(">",i,"\n",sep="") #adjust space between > and number using sep
  aa<- generate_protein(i)
  cat(aa)
  cat("\n")
}
```

```
>6
LKIIAK
>7
YDEYQSF
>8
EYMHMYCY
>9
IGYIVPIHA
>10
CVAQVQCIEL
>11
FGSQMKLQNWL
>12
LTVCHTCLDIPH
```

A third, and better, way to solve this is to use the `apply()` family of functions, specifically the `sapply()` function in this case

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

```
[1] "FTAYWM"       "HHQIEMM"      "CFTDKSWV"     "AIQQTQTKH"    "MLTACPNHRQ"  
[6] "GDKYRPTHNNQ"  "LNINKSSEVMMW"
```