Class6

Ebru Robinson

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 **arguments** (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

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

Now we can call this function:

```
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.

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

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

I want the a 1 elemnt long character vectro that looks this “ACCACT” not “A” “C” “C” “A” “C” “T”

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

```
[1] "GACGAGGCGAATCAGCGGCGTCTGACGCTTGGTCAGAAGGTCGCTACATT"
```

Turn this into my first function

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

Test it:

```
generate_dna(60)
```

```
[1] "TCTCGTTATATCTCGCGACTCTGTCTAAGTAAAGGACTAGTCGGGCCAGTCGAGCTGTTT"
```

```
fasta <- FALSE
if(fasta) {
  cat("HELLO You")
}
```

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

```
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)
}
}
```

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

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

A protein genearting function

```
generate_protein <- function(size = 50, fasta = TRUE) {
  amino_acids <- c("A", "R", "N", "D", "C", "Q", "E", "G",
                   "H", "I", "L", "K", "M", "F", "P", "S",
                   "T", "W", "Y", "V")

  v <- sample(amino_acids, size = size, replace = TRUE)
  s <- paste(v, collapse = "")

  if (fasta) {
    return(s)
  } else {
    return(v)
  }
}
```

```
generate_protein(6)
```

```
[1] "SIFAWT"
```

Use our new `generate_prtein()` function to make random protein sequences of length 6 to 12 (i.e. one length 7, etc. up to length 12) one way to do is "brute force"

```
generate_protein(6)
```

```
[1] "AETTTE"
```

```
generate_protein(7)
```

```
[1] "ITDYWDA"
```

```
generate_protein(8)
```

```
[1] "AWPWSVLM"
```

```
generate_protein(9)
```

```
[1] "SYSFMHNMN"
```

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

```
lengths <- 6:12
lengths
```

```
[1]  6  7  8  9 10 11 12
```

```
for(i in lengths) {
 cat(">",i,"\n",sep="")
  aa <- generate_protein(i)
  cat(aa)
  cat("\n")
}
```

```
>6
TVFVTW
>7
CLTYRPW
>8
IAHTPYLC
>9
KFMSDRDRL
>10
VPEFSPDTCL
>11
TDNNPFWADYE
>12
LYMGQQCFNTEV
```

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

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

```
[1] "HVGPMD"       "VLSFVEL"      "NHHFHSYC"     "SKWPMSKCR"    "CRFGVIEICN"
[6] "DLSDRGGEEDE"  "RGVLQNTMERFC"
```