Class6:R functions

Zixuan Zeng (A16142927)

Table of contents

All functions in R have at least 3 things:

- A **name**, we oucj this and use it to call our function, -Input **arguments** (there can be multiple) The **body** lines o 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 the function:

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

```
[1] 110 110
```

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

```
[1] 110
```

A second function

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

THe `sample()` function can be helpful here.

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

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

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

```
[1] "ACAAGTTGCCACGAGTGATCGGTCAGTCACCAAGTTTTCTCAATGTTCAG"
```

Turn this into generate_DNA

```
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] "TATTCATCCTACTCGGTAGCCCGGGCCGCCCCTAAATTAGGCGATGTTTGTTGTTCGGCA"
```

```
fasta <- FALSE
if(fasta){
  cat("HELLO You!")
} else{
  cat("No you dont!")
}
```

```
No you dont!
```

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

```
generate_DNA <- 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)
}
}
```

Test:

```
generate_DNA(60,fasta=FALSE)
```

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

a protein generating function

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

if(fasta){
  return(s)
} else{
  return(V)
}
}
generate_protein(6)
```

```
[1] "TSQLEK"
```

Use our new `generate_protein`function to make random protein sequences of length 6 to 12 (i.e. one length 6, one length 7, up to length 12)

brute force:

```
generate_protein(6)
```

```
[1] "KMHEAH"
```

```
generate_protein(7)
```

```
[1] "FGIRDYI"
```

```
generate_protein(8)
```

```
[1] "DWVFRTQG"
```

A second way is to use a `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
VKQVEE
>7
LHEVARS
>8
IDDTHDPR
>9
TQDVNGILT
>10
WPMAALIWFA
>11
QDPKWFFCSAE
>12
SYHCHVITKDIN
```

A thrid, 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] "VDNPFT"       "DACGRWW"      "LWTKCGCD"     "CDEIQLPRW"    "ECCYSLIFDC"
[6] "FVITHHVLHMS"  "VHQRIQTRRGNL"
```