class06

Rachel Field

Table of contents

#functions have names,input arguments, and body

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

#call the function

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

```
[1] 110
```

#second practice function

```
sample (c("A", "C", "G","T"), size=5, replace = TRUE)
```

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

##Turn this into my first function

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

##Test it

```
generate_data(60)
```

```
[1] "GTAAGGAAGTCGGGCGTCATATCGACGGGGTCGGTTTGGTGCCCATTCGACGGTAGGATG"
```

```
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_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")
} else {
  cat("No you dont!")
}
}
```

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

```
HELLO You
```

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

```
No you dont!
```

```
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=TRUE)
```

```
[1] "TCTCATAATC"
```

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

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

#notetoself: sample() is a function that randomly selects elements from a vector. c("A", "C", "G", "T") is the vector from which we're sampling — representing the four DNA bases. size=size means you're sampling size number of bases (default is 50). replace=TRUE means that each selection is with replacement, so the same base can appear multiple times. #paste() combines elements of a character vector into a single string, collapse=" " tells it not to add any separator — so the bases are just stuck together.

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

##Use our new `generate_protein()` function to make random sequences of length 6 to 12 (i.e. one length 6, one length 7, one length 8, etc).

One way to do this is brute force.

```
generate_protein(6)
```

Or use a for loop

```
lengths<-6:12
for(i in lengths) {
  cat(i)
  cat("\n")
}
```

```
6
7
8
9
10
11
12
```

```
lengths<-6:12
for(i in lengths) {
  cat(">", i, sep= "")
  aa<-generate_protein(i)
  cat(aa)
  cat("\n")
}
```

```
>6CSVVAA
>7KRERFNA
>8WNNYWKLY
>9ESEAWGYDG
>10VLKAVLNDYG
>11GIVGATTWFRG
>12RNITMMQFPSST
```

##sapply(vector or list, function)

```
generate_protein<-function(size=50, fasta=TRUE) {
  aa<-c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I", "L", "K", "M", "F", "P", "S", "T",
  v<- sample(aa, size=size, replace=TRUE)
  s<-paste(v, collapse="")
}
sapply(6:12, generate_protein)
```

```
[1] "RLFKEF"       "GNRCIWM"      "STPYCATD"     "CPAMEKPGV"    "ADIWQQKLAT"
[6] "CTYLGPDTRKG"  "GGNDMGFHLEGV"
```