Class 6: R Functions

Allen X. (A16897142)

Table of contents

Basics to Functions

Here we have our first fun function or (FUN-ctions) to help add numbers:

Every R function has three things:

- name (we will pick this)
- input arguments (there can be many that are separated with a comma)
- the body (R code does this work)

```r
add <- function(x, y=100, z=0){
  x + y + z
}
```

I can now use this function whenever, but you need to run the code chunk first:

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

```
[1] 101
```

```r
add(c(1,2,3,4))
```

```
[1] 101 102 103 104
```

```r
add(1)
```

```
[1] 101
```

Functions can have "required" input arguments and "optional" input arguments. The optional arguments are defined with an equals default value and kind of serves as a fallback value. For example (y=10).

```r
add(x=1,y=100,z=10)
```

```
[1] 111
```

> Write a function to return a DNA sequence of a user specified in length. Use the function `generate_dna()`

```r
#generate_dna <- function(size=5)]{}

students <- (c("jeff","jeremy","peter"))

sample(students, size =1, replace=TRUE)
```

```
[1] "jeremy"
```

The code above is used when you want to generally select from a sample size. Replace argument allows you to redo selections.

Generate DNA sequence

Now we will work with bases and not students:

```r
bases <- c("A","C","G","T")
sample (bases, size=10, replace=TRUE)
```

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

This 'snippet' of DNA is what we want and will serve as the body of the function we want to reuse.

```r
generate_dna <- function(size=5){
  bases <- c("A","C","G","T")
sample (bases, size=size, replace=TRUE)
}
```

```r
generate_dna(100)
```

```
  [1] "G" "A" "G" "C" "C" "T" "G" "T" "G" "C" "G" "A" "C" "T" "C" "G" "G" "A"
 [19] "T" "T" "C" "T" "A" "C" "G" "A" "T" "A" "C" "G" "G" "C" "A" "C" "T" "C"
 [37] "A" "T" "C" "A" "G" "C" "C" "A" "A" "G" "C" "A" "C" "T" "G" "A" "A" "T"
 [55] "C" "C" "A" "A" "C" "T" "C" "A" "C" "A" "G" "C" "G" "G" "C" "T" "A" "G"
 [73] "G" "T" "C" "T" "C" "G" "A" "C" "A" "G" "A" "T" "C" "C" "A" "G" "A" "G"
 [91] "G" "T" "T" "T" "T" "T" "C" "T" "T" "T"
```

Now I want a one element vector sequence like "ATGACTACC", and not split up with quotes.

```r
generate_dna <- function(size=5, together=TRUE){
  bases <- c("A","C","G","T")
  sequence <- sample (bases, size=size, replace=TRUE, )
  if(together){
  sequence <- (paste(sequence,collapse=""))
  }
  return(sequence)
}
```

```r
generate_dna(5)
```

```
[1] "CCCTC"
```

```r
generate_dna(together = F)
```

```
[1] "C" "T" "T" "T" "C"
```

Generate Protein Functions

If you need the set of 20 amino acids for homework and class, you download the **bio3d** package

```r
aa <- bio3d::aa.table$aa1[1:20]
```

> Write a protein sequence generating function that will return sequences of a specified length of 5.

```r
generate_aa <- function(size=5, together=T){
  ##Get the 20 amino acids as a vector through this way:
  aa <- bio3d::aa.table$aa1[1:20]
  sequence <- sample (aa, size=size, replace=TRUE, )
  ## This gives us a string without ""
  if(together){
  sequence <- (paste(sequence,collapse=""))
  }
  return(sequence)
}
```

```r
generate_aa(5)
```

```
[1] "MDLTL"
```

> Generate random protein sequences of length 6 to 12 amino acids.

```r
generate_aa()
```

```
[1] "PLQIH"
```

We fix this initial error message by adding to the function body code. Using the R **apply** family of utility functions.

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

```
[1] "CNLACC"       "GQPCHQD"      "IGDRFSLH"     "PYMETPGFE"    "NKFNKISIAR"
[6] "TTKHTYSNKDR"  "LCMALIRKRCDW"
```

Trying to get FASTA format output

```r
ans <- sapply(6:12, generate_aa)
ans
```

```
[1] "DMEWCI"        "KFCYFHS"       "WHWIQRSW"      "PAKYHILQK"     "QETTSDTKSQ"
[6] "CMNGTCFADYT"   "SHGHLAQVNKKE"
```

```r
cat(ans,sep="\n")
```

```
DMEWCI
KFCYFHS
WHWIQRSW
PAKYHILQK
QETTSDTKSQ
CMNGTCFADYT
SHGHLAQVNKKE
```

We want to thing to look like:

```
>ID.6
AAAAAA
>ID.7
AAAAAAA
>ID.8
AAAAAAAA
ETC.
```

Functions `paste()` and `cat()` will help us:

```r
cat(paste(">ID.",6:12, "\n", ans,sep=""), sep="\n")
```

```
>ID.6
DMEWCI
>ID.7
KFCYFHS
>ID.8
WHWIQRSW
>ID.9
PAKYHILQK
>ID.10
QETTSDTKSQ
>ID.11
CMNGTCFADYT
>ID.12
SHGHLAQVNKKE
```

```r
id.line <- paste(">ID.",6:12,sep="")
seq.line <- paste(id.line,ans,sep="\n")
cat(seq.line,sep="\n", file="myseq.fa")
```

> Can you determine if these sequences can be found in nature or not? Why or why not?

BLASTp your FASTA format sequences against NR and found that the aa sequence of 6,7,8 are found in nature as they have 100% identity and coverage, but the 9, 10, 11, 12 base pair sequences are unique as they do not have 100% identity and coverage.