# Functions in R

Emily Chase (PID: A14656894)

## Table of contents



### Write a function that adds numbers

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

}
```

Call the function

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

```
[1] 110
```

### Another function

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

The `sample` function can be helpful here. It samples randomly from an input vector.

```
sample(c("A", "C", "G", "T"), size=5, replace=TRUE) # can only have size>n if replace=TRUE
```

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

`sample` gives us a vector, but what if we want a fasta format instead? ie a 1 element long character vector

```
s <- sample(c("A", "C", "G", "T"), size=5, replace=TRUE) # can only have size>n if replace=TR
paste(s, collapse="")
```

```
[1] "TTACA"
```

We can use `paste` (kinda like `join` in python). `paste` has two arguments that are new to me though:

1. `collapse` = _____ ~ determines what will separate **ELEMENTS** in a vector
2. `sep` = _____ ~ determines what will separate **arguments**. you can give `paste` multiple vectors and it will combine them pair-wise, and you can tell it what to separate those pairs by.

### Conditionals interlude

```
fasta <- TRUE
if (fasta){
  cat("HELLO WORLD!")
} else {
  cat("NOPE")
}
```

```
HELLO WORLD!
```

```
# what's cat()

print("HI")
```

```
[1] "HI"
```

```
print(c("HI", "BYE"))
```

```
[1] "HI"  "BYE"
```

```
cat(c("HI", "BYE"))
```

```
HI BYE
```

### Now integrate it into functions

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", "G", "C","T"), size=size, replace=TRUE)


  if (fasta){
    ans <- paste(v, collapse="")
    return(ans)
  } else {
    return(v)
  }

}

generate_fasta(fasta=TRUE)
```

```
[1] "GGCCTGCACTTTTGTTATTGTGTCACCGTGGCGGATCGGCTTGCAGTTAA"
```

```
generate_fasta(fasta=FALSE)
```

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

## Design a protein generating function

```
generate_pfasta <- function(size=50, fasta=TRUE){
  v <- sample(c("A", "R", "N", "D", "C", "Q", "E", "G", "H", "I", "L", "K", "M", "F", "P", "S
  if (fasta){
    ans <- paste(v, collapse="")
    return(ans)
  } else {
    return(v)
  }

}

generate_pfasta(fasta=TRUE)
```

```
[1] "HRQPVDLCSGMWYNAPLDHPLHEYMMVSWVDYPGGPMCKAWKCLVGVDPP"
```

```
generate_pfasta(fasta=FALSE)
```

```
 [1] "S" "S" "M" "Q" "G" "D" "W" "F" "S" "A" "H" "G" "Y" "M" "M" "T" "I" "Q" "W"
[20] "C" "Q" "P" "K" "A" "Y" "L" "D" "F" "T" "M" "G" "C" "E" "M" "Y" "G" "H" "N"
[39] "S" "D" "K" "L" "S" "V" "N" "N" "I" "S" "C" "T"
```

```
generate_pfasta(6)
```

```
[1] "SNLGAY"
```

### Generate sequences length between 6 and 12

Use our new `generate_pfasta()` function to make random protein sequences of length 6 to 12 (ie one length 6, one length 7, up to len 12)

```
# brute force
generate_pfasta(6)
```

```
[1] "SLINDY"
```

```
generate_pfasta(7)
```

```
[1] "KLHESVI"
```

```
generate_pfasta(8)
```

```
[1] "WDFNKAVT"
```

```
# using a for loop

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

```
>6
WWRPKM
>7
CGHDFGQ
>8
DQQRSDVW
>9
PTGEDIPVW
>10
HMMASMPISD
>11
FEMIMPPSSEK
>12
SYSCYHATLFPF
```

```
# more efficient

sapply(lengths,generate_pfasta)
```

```
[1] "CVNCSR"       "NIIKEFE"      "PCELSMGI"     "QQYVGHLTA"    "QIKMPSERII"
[6] "IPNRLPEGEKS"  "IDINFWLESINL"
```

## Search in BLAST