# Homework Assignment 1

August 30, 2020

## 1 Combining with `c`

Single-word answers are fine for 1A-1H:

A. After you run `wal = c(c(1, 2, 3), "string")`, what is the `class` of `wal`?
B. What is the `class` of `wal[[1]]`?
C. After you run `rus = c(list(1, 2, 3), "string")`, what is the `class` or `rus`?
D. What is the `class` of `rus[[1]]`?
E. What is the `class` of `c(wal, rus)`?
F. What is the `length` or `c(wal, rus)`?
G. What is the `length` of `c(list(wal), rus)`?
H. What is the `length` of `c(list(wal), rus)[[1]]`?
I. What is the `length` of `c(list(wal), rus)[1]`? Why is this different from your last answer? (*Hint:* Compare the `class`es of the two objects; consider what single brackets are used for in an expression like `c(list(wal), rus)[1:3]`; and note that in R the number `1` is the same thing as the vector `1:1`.)

## 2 Overlapping intervals

Let's say you have a piece of linear DNA containing (at least) two genes. Assume that for each of the two genes you are given the start and end coordinates—two integers specifying (first) the lowest-numbered base and (second) the highest-numbered base bracketing the region of DNA encoding the gene in question—in the form of an R vector. We'll call the 2-element vector encoding the interval for gene 1 `g` and the 2-element vector encoding the interval for gene 2 `h`.

A. Write a function `overlap` taking `g` and `h` as arguments which returns the logical value `TRUE` if gene 1 overlaps gene 2 and `FALSE` otherwise.
B. What does `overlap(g=c(200, 400), h=c(100, 250))` return?
C. What about `overlap(c(500, 600), c(800, 900))`?

## 3 Fun with functions

Start with a simple function, `f0`, which simply adds 1 to its argument:

```
> f0 = function(x) {x+1}
```

We're going to build some other functions based on this one using a function `functionalPower`:

```
> functionalPower = function(f, power) {
      fRepeatedPowerTimes = function(x) {
          if (power == 0) {
              return(x)
          } else {
              for (i in 1:power) {
                  x = f(x)
              }
          }
          return(x)
      }
      return(fRepeatedPowerTimes)
  }
```

`functionalPower` takes as arguments a function `f` and an nonnegative integer `power`, internally builds a function `fRepeatedPowerTimes` which, when applied to an argument `x`, returns

$$\underbrace{\texttt{f}(\texttt{f}(\cdots\texttt{f}(\texttt{x})\cdots))}_{\texttt{power} \text{ repetitions of } \texttt{f}} \tag{1}$$

Notice that the function returned by `functionalPower` must in some sense "remember" both the function `f` it is based on and the integer `power` telling it how many times to repeat `f`. This makes any function `fRepeatedPowerTimes` created by `functionalPower` a *closure* in programming-speak, with the recorded variables `f` and `power` enclosed within the *environment* of `fRepeatedPowerTimes`. (Not all programming languages support closures, but R and Python both do!)

### 3.1 Powers of `f0`

Let's put `functionalPower` and `f0` to use to make a new function `f1`:

```
> f1 = function(x) {
      return(function(y) {
          f0_to_y = functionalPower(f0, y)
          return(f0_to_y(x))
      })
  }
```

A. What is `f1(2)(3)`?
B. What is `f1(3)(4)`?
C. More generally, what does the function `f1(2)` return given argument `y`?
D. Define `` `%op1%` = function(x, y) {f1(x)(y)} ``. What does `x %op1% y` represent?

### 3.2 Powers of `f1`

Now define

```
> f2 = function(x) {
      return(function(y) {
          f1ofx_to_yminus1 = functionalPower(f1(x), y-1)
          return(f1ofx_to_yminus1(x))
      })
  }
```

A. What is `f2(2)(3)`?
B. What is `f2(3)(4)`?
C. More generally, what does the function `f2(2)` return given argument `y`?
D. Define `` `%op2%` = function(x, y) {f2(x)(y)} ``. What does `x %op2% y` represent?

### 3.3 Powers of `f2`

Let's try

```
> f3 = function(x) {
      return(function(y) {
          f2ofx_to_yminus1 = functionalPower(f2(x), y-1)
          return(f2ofx_to_yminus1(x))
      })
  }
```

A. What is `f3(2)(3)`?
B. What is `f3(3)(4)`?
C. More generally, what does the function `f3(2)` return given argument `y`?
D. Define `` `%op3%` = function(x, y) {f3(x)(y)} ``. What does `x %op3% y` represent?