Iteration with loops

Kaitlyn Watson Group 9

M4 ICA2

Function `fib()`

Introduction

The Fibonacci sequence is the sequence $\{F_n : n \geq 1\}$ of numbers defined by $F_1 = 0, F_2 = 1$ and $F_n = F_{n-2} + F_{n-1}$ for $n \geq 3$. For example, the first seven Fibonacci numbers are

$$
\begin{aligned}
F_1 &= 0 \\
F_2 &= 1 \\
F_3 &= 0 + 1 = 1 \\
F_4 &= 1 + 1 = 2 \\
F_5 &= 1 + 2 = 3 \\
F_6 &= 2 + 3 = 5 \\
F_7 &= 3 + 5 = 8
\end{aligned}
$$

Part a

`fib()`

Write a function, `fib()`, which uses a `for` loop. Function `fib()` should take one argument, `n`, and return the first `n` Fibonacci numbers in a vector. See below for some examples.

```
fib <-function(n) {                      #creates function
  fib <- c(0,1)                            # first 2 numbers in the sequence
  if(n <= 2){                              # checks the index is less than or equal to 2
  fib<-fib[1:n]                              #Gives position of the sequence
}
if(n > 2 ){                              #checks if greater than 2
     for (i in 3:n) {
          fib[i]<- fib[i-2]+fib[i-1]
       }
     }
  return(fib)
}
```

```
fib(n = 1)
fib(n = 2)
fib(n = 7)
fib(n = 10)
```

Part b

`fib_1()`

Modify `fib()` so that it uses a `while` loop instead of a `for` loop. Call this new function `fib_1()`. Test that your function works with the following examples below.

```
fib_1 <-function(n) {                    #creates function
  fib_1 <- c(0,1)                          # first 2 numbers in the sequence
  if(n <= 2){                          # checks the index is less than or equal to 2
  fib_1<-fib_1[1:n]                          #Gives position of the sequence
}
if(n > 2 ){                                #checks if greater than 2
     i<- 3
     while (i<=n) {
          fib_1[i]<- fib_1[i-2]+fib_1[i-1]
          i<- i+1
       }
     }
  return(fib_1)
}
```

```
fib_1(n = 1)
fib_1(n = 2)
fib_1(n = 7)
fib_1(n = 10)
```

Function `all.primes()`

Introduction

Your goal is to write a function `all.primes()` that will return all possible prime numbers among the first `n` positive integers. For example, among the integers 1 through 10, there are prime numbers: 2, 3, 5, 7. Recall that a prime number is a whole number greater than 1 whose only factors are 1 and itself.

To get started, proceed to step 1.

Step 1

`is.prime()`

R has built-in functions that perform logical tests. For example, `is.numeric()` and `is.character()` performs a logical test if an object is numeric and if an object is of type character, respectively. Write a function called `is.prime()` that tests if a numeric vector of length 1 is a prime number or not. The function should return a single `TRUE` or `FALSE` value. See below for some examples.

```
is.prime <- function(x) {
   if (x == 2) {
      TRUE
   } else if (any(x %% 2:(x-1) == 0)) {
      FALSE
   } else {
      TRUE
   }
}
```

```
is.prime(x = 2)
is.prime(x = 10)
is.prime(x = 13)
```

Step 2

`all.primes()`

Write a function `all.primes()` that has one argument, `n`, and returns all possible prime numbers among the first `n` positive integers. See below for some examples.

```
all.primes = function(n){

primes <- numeric()
  j <- 1
  for (i in 2:n) {
    if (is.prime(i)) {
      primes[j] <- i
      j <- j + 1
    }
  }
  return(primes)
  }
```

```
all.primes(n = 10)
all.primes(n = 21)
all.primes(n = 2)
```

Reference: Stack Overflow Code For Prime Number Function https://stackoverflow.com/questions/19767408/prime-number-function-in-r Accessed on October 20, 2021

Reference: Pratim Guhaniyogi For All prime code chunk Accessed on October 20,