Introduction to the R Statistical Computing Environment
R Programming I: Exercise

John Fox
(McMaster University)
ICPSR

2021

* *Loop versus recursion:* Named after a famous medieval Italian mathematician, Fibonacci numbers are an integer sequence F_n defined for $n = 1, 2, \ldots$ as

$$\begin{aligned} F_1 &= F_2 = 1 \\ F_n &= F_{n-1} + F_{n-2} \texttt{ for } n > 2 \end{aligned}$$

This definition leads straightforwardly to a recursive function to compute Fibonacci numbers; write such as function, `fib0(n)`. Verify that your function works, as follows:

```
> sapply(1:10, fib0)
 [1] 1 1 2 3 5 8 13 21 34 55
```

The largest Fibonnaci number that can be represented exactly as a double-precision floating-point number (on most computers) is $F_{78} = 8,944,394,323,791,464$, but `fib0` would take a very, very, very long time to compute this number. Let's consider another approach to the computation, which is to do it iteratively:

```
fib1 <- function(n){
    if (n <= 2) return(1)
    last.minus.1 <- 1
    last.minus.2 <- 1
    for (i in 3:n){
        save <- last.minus.1
        last.minus.1 <- last.minus.1 + last.minus.2
        last.minus.2 <- save
    }
    last.minus.1
}
```

Compare the time required to compute `fib0(35)` versus `fib1(35)`. Also check that `fib1(78)` gives you the right answer. To suppress scientific notation, you can set `options(scipen=10)`.

Finally, although Fibonacci numbers are defined by the recurrence relation above, they may also be computed directly by Binet's formula, as

$$F_n = \left[\frac{\left(\frac{1+\sqrt{5}}{2}\right)^n}{\sqrt{5}} \right]$$

where the square brackets represent rounding to the nearest integer. Because of rounding errors on a computer using double-precision floating-point arithmetic, this result produces an accurate answer only up to $F_{70} = 190,392,490,709,135$. Veryify that this is the case by programming the formula as `fib2(n)` and checking `fib1(70)` and `fib1(71)` versus `fib2(70)` and `fib2(71)`.