# Lista 4 - Introdução a Análise de Dados Funções & Loops

# Guilherme Masuko

# April 2023

## Questão 1

Crie dois loops. O primeiro deverá printar na tela os valores de 1 à 10 fazendo uso do for. O segundo deverá printar no console os valores de 1 à 5 usando o comando while.

```
rm(list=ls())

# declarando a variável a ser utilizada
x <- 0

for (i in 1:11) {
    print(x)
    # precisamos ir incrementando 1 a cada iteração
    x <- x + 1
}

# declarando a variável a ser utilizada
x <- 0

while (x <= 5) {
    print(x)
    # precisamos ir incrementando 1 a cada iteração
    x <- x + 1
}</pre>
```

Crie duas funções, ate\_n\_for e ate\_n\_while, que recebam um parâmetro n e executam os comandos que fizemos na questão anterior. Isso é, a primeira função deverá printar na tela os valores de 1 à n fazendo uso do for. A segunda deverá printar no console os valores de 1 à n usando o comando while.

```
rm(list=ls())
# funções
ate_n_for <- function (n) {</pre>
 # declarando a variável a ser utilizada
 x <- 0
 for (i in 1:(n+1)) {
   print(x)
   # precisamos ir incrementando 1 a cada iteração
   x < -x + 1
 }
}
# teste
ate_n_for(15)
ate_n_while <- function (n) {</pre>
 # declarando a variável a ser utilizada
 x <- 0
 while (x \le n) {
  print(x)
   # precisamos ir incrementando 1 a cada iteração
   x < - x + 1
 }
# teste
```

Crie um loop que imprima no console a tabuada de 1 à 10.

# Solução

```
rm(list=ls())

# loop para o primeiro elemento do produto
for (x1 in 1:10) {
    # loop para o segundo elemento do produto
    for (x2 in 1:10) {
        # resultado do produto
        y = x1 * x2
        print(paste(x1,'x',x2,'=',y))
    }
}
```

# Questão 4

a) Faça um loop para fazer o somatório de um vetor que vai de 1 à 10.

$$1 + 2 + 3 + \dots + 10$$

```
rm(list=ls())

# declarando uma variável que guardará o valor da soma
soma = 0

# loop
for (i in 1:10) {
   soma = soma + i
}

print(soma)
```

**b)** Agora crie uma função que recebe o valor n e retorna o valor da soma.

$$1+2+3+\ldots+n$$

# Solução

```
rm(list=ls())

somaSequencia <- function(n) {
    # declarando uma variável que guardará o valor da soma
    soma <- 0
    # loop
    for (i in 1:n) {
        soma = soma + i
    }
    return(soma)
}

# teste

somaSequencia(18)</pre>
```

# Questão 5

a) Faça um loop que realize o cálculo do fatorial de 10.

$$10! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot 10$$

```
rm(list=ls())
# declarando uma variável que guardará o valor do produtório
fatorial = 1
```

```
# loop
for (i in 1:10) {
  fatorial = fatorial*i
}
print(fatorial)
```

**b)** Agora crie uma função que recebe o valor n e retorne o valor do fatorial.

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$$

# Solução

```
rm(list=ls())

calculaFatorial <- function (n) {
  # declarando uma variável que guardará o valor do produtório
  fatorial = 1

  # loop
  for (i in 1:n) {
    fatorial = fatorial*i
  }
  return(fatorial)
}</pre>
```

# Questão 6

 a) Use o loop while para investigar o número de termos necessários antes que o produto

$$1 \cdot 2 \cdot 3 \cdot \dots$$

alcance mais que um milhão.

# Solução

```
rm(list=ls())

# declarando variáveis

# variável que guardará o valor do produtório
x <- 1

# variável que guardará o número de termos
count <- 0

# loop
while (x <= 1000000) {
    count <- count + 1

    # a cada iteração multiplica-se o valor agregado pelo número
        à frente do que se multiplicou na iteração passada
    x = x*(count+1)
}

print(count)</pre>
```

b) Agora faça uma função que receba o valor n e retorne o número de termos necessários antes que o produto acima alcance mais que n.

```
rm(list=ls())

# função

termos_necessarios <- function(n) {
    # declarando variáveis

# variável que guardará o valor do produtório
    x <- 1
    # variável que guardará o número de termos
    count <- 0

while (x <= n) {</pre>
```

A sequência de Fibonacci<sup>1</sup> é uma sequência de números inteiros, começando normalmente por 0 e 1, na qual o termo subsequente corresponde à soma dos dois anteriores.

Os números que compõem a sequência de Fibonacci são:

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, \dots
```

A sequência é definida recursivamente pela fórmula abaixo.

$$F_n = F_{n-1} + F_{n-2}$$

e valores iniciais  $F_1 = 0$  e  $F_2 = 1$ .

Notação: A notação  $(F_n)_{n\in A}$  é usada para denotar a sequência F, cujos índices são tomados no conjunto A. Quando o conjunto dos índices A está subentendido, normalmente escrevemos  $(F_n)_n$  ou, simplesmente,  $(F_n)$ . Por extenso, escrevemos  $(F_n)_n = (F_1, F_2, F_3, \ldots)$ . Observamos, ainda, que as notações  $\{F_n\}_{n=1}^{\infty}$  e  $\{F_n\}$  também são encontradas.

a) Crie um vetor contendo os primeiros 50 valores da sequência de Fibonacci. Utilize o comando for para adicionar novos elementos ao vetor.

<sup>&</sup>lt;sup>1</sup><a href="https://pt.wikipedia.org/wiki/Sequ%C3%AAncia\_de\_Fibonacci">https://pt.wikipedia.org/wiki/Sequ%C3%AAncia\_de\_Fibonacci</a>

```
rm(list=ls())

# criando um vetor que guardará a sequência de Fibonacci com os
    dois primeiros elementos da sequência

fib = c(0,1)

# o loop começa no índice 3 porque já definimos os dois
    primeiros elementos

for (i in 3:50) {

    # cada elemento subsequente é definimo pela soma dos dois
    anteriores
    fib[i] = fib[i-1] + fib[i-2]
}

print(fib)
```

b) Crie uma função que recebe como parâmetro o valor n. A função deve retornar um vetor contendo os primeiros n valores da sequência de Fibonacci utilizando o programa feito a partir do comando for.

```
rm(list=ls())

# função

fibonacciFor <- function (n) {
    # criando um vetor que guardará a sequência de Fibonacci com
    os dois primeiros elementos da sequência
    fib = c(0,1)

# o loop começa no índice 3 porque já definimos os dois
    primeiros elementos

for (i in 3:n) {

    # cada elemento subsequente é definimo pela soma dos dois
    anteriores</pre>
```

```
fib[i] = fib[i-1] + fib[i-2]
}
return(fib)

# teste

fibonacciFor(10)
```

a) Crie um vetor contendo os primeiros 50 valores da sequência de Fibonacci. Utilize o comando while para adicionar novos elementos ao vetor.

## Solução

```
rm(list=ls())

# criando um vetor que guardará a sequência de Fibonacci com os
    dois primeiros elementos da sequência
fib = c(0,1)

# criando uma variável que conta a quantidade de elementos na
    sequência
i <- 2
while (i < 50) {
    # a cada iteração a sequência ganha um novo elemento
    i <- i + 1

# cada elemento subsequente é definimo pela soma dos dois
    anteriores
fib[i] = fib[i-1] + fib[i-2]
}

print(fib)</pre>
```

b) Crie uma função que recebe como parâmetro o valor n. A função deve retornar um vetor contendo os primeiros n valores da sequência de Fibonacci utilizando o programa feito a partir do comando while.

```
rm(list=ls())
fibonacciWhile <- function (n) {</pre>
 # criando um vetor que guardará a sequência de Fibonacci com
    os dois primeiros elementos da sequência
 fib = c(0,1)
 # criando uma variável que conta a quantidade de elementos na
    sequência
 i <- 2
 while (i < n) {
   # a cada iteração a sequência ganha um novo elemento
  i <- i + 1
   # cada elemento subsequente é definimo pela soma dos dois
      anteriores
  fib[i] = fib[i-1] + fib[i-2]
 }
 return(fib)
# teste
fibonacciWhile(10)
```