Capítulo 03: Noções básicas de programação

Igo da Costa Andrade

2023-11-08

Exercícios

3. A função `nchar` informa quantos caracteres um vetor de caracteres possui. Escreva uma linha de código que atribua ao objeto `new_names` a abreviatura do estado quando o nome do estado tiver mais de 8 caracteres.

```
data("murders")

new_names <- ifelse(nchar(murders$state) <= 8, murders$state, murders$abb)

new_names
```

```
##  [1] "Alabama"  "Alaska"   "Arizona"  "Arkansas" "CA"       "Colorado"
##  [7] "CT"       "Delaware" "DC"       "Florida"  "Georgia"  "Hawaii"
## [13] "Idaho"    "Illinois" "Indiana"  "Iowa"     "Kansas"   "Kentucky"
## [19] "LA"       "Maine"    "Maryland" "MA"       "Michigan" "MN"
## [25] "MS"       "Missouri" "Montana"  "Nebraska" "Nevada"   "NH"
## [31] "NJ"       "NM"       "New York" "NC"       "ND"       "Ohio"
## [37] "Oklahoma" "Oregon"   "PA"       "RI"       "SC"       "SD"
## [43] "TN"       "Texas"    "Utah"     "Vermont"  "Virginia" "WA"
## [49] "WV"       "WI"       "Wyoming"
```

4. Crie uma função `sum_n` que, para qualquer valor, digamos n, calcula a soma dos inteiros de 1 a n (inclusive). Use a função para determinar a soma dos números inteiros de 1 a 5.000.

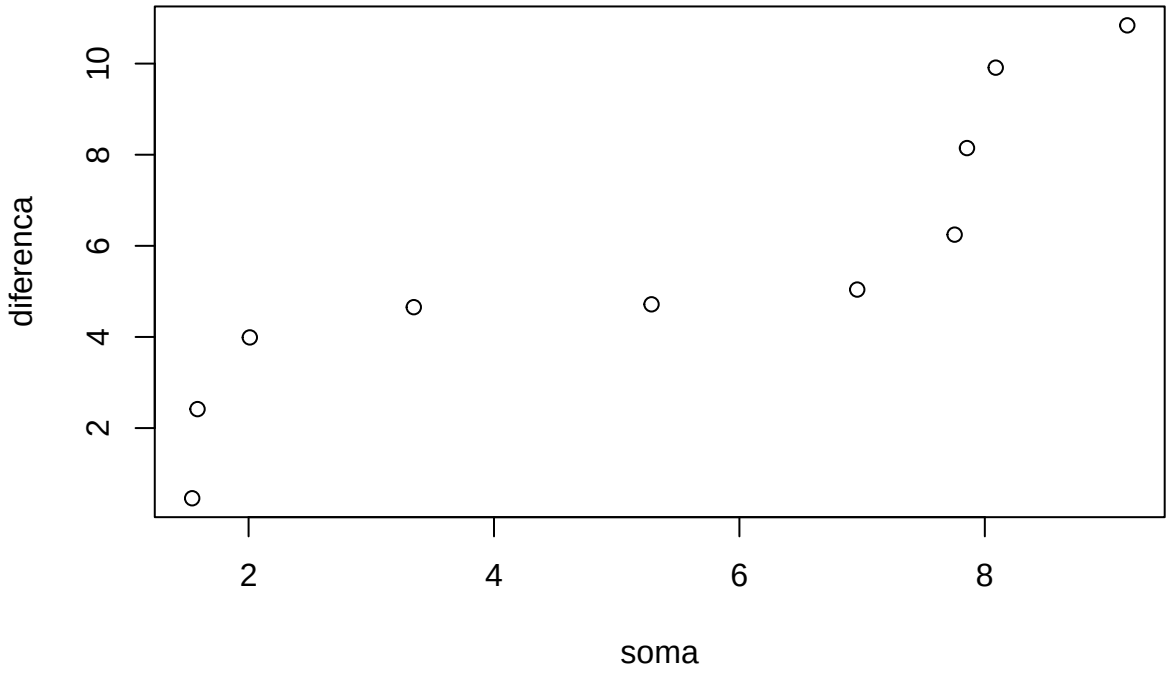
```
sum_n <- function(n) {
  s <- 0
  for (i in 1:n) {
    s <- s + i
  }
  s
}

sum_n(5000)
```

```
## [1] 12502500
```

5. Crie uma função `altman_plot` que receba dois argumentos `x` e `y`, e represente graficamente a diferença em relação à soma.

```
altman_plot <- function(x, y) {
  if (length(x)==length(y)) {
    soma <- x + y
    diferenca <- x - y
    plot(soma, diferenca)
  } else {
    print('Vetores devem possir mesmo comprimento!')
```

```
  }
}

x <- 1:10
y <- cos(x)


altman_plot(x, y)
```



6. Após executar o código abaixo, qual o valor de `x`?

```
x <- 3

my_func <- function(y) {
  x <- 5
  y+5
}
```

O valor de `x` continua igual a 3.

7. Escreva uma função `compute_s_n` que para qualquer n dado calcule a soma

$$S_n = 1^2 + 2^2 + 3^2 + \cdots + n^2.$$

Informe o valor da soma quando $n = 10$.

```
compute_s_n <- function(n) {
  sum((1:n)^2)
}

compute_s_n(10)
```

```
## [1] 385
```

8. Defina um vetor numérico vazio `s_n` de tamanho 25 usando `s_n <- vector("numeric", 25)` e armazene os resultados de $S_1, S_2, \cdots, S_{25}$ usando um *loop for*.

```r
s_n <- vector("numeric", 25)

for (i in 1:25) {
  s_n[i] <- compute_s_n(i)
}

s_n
```

```
##  [1]    1    5   14   30   55   91  140  204  285  385  506  650  819 1015 1240
## [16] 1496 1785 2109 2470 2870 3311 3795 4324 4900 5525
```

9. Repita o exercício 8, mas desta vez use `sapply`.

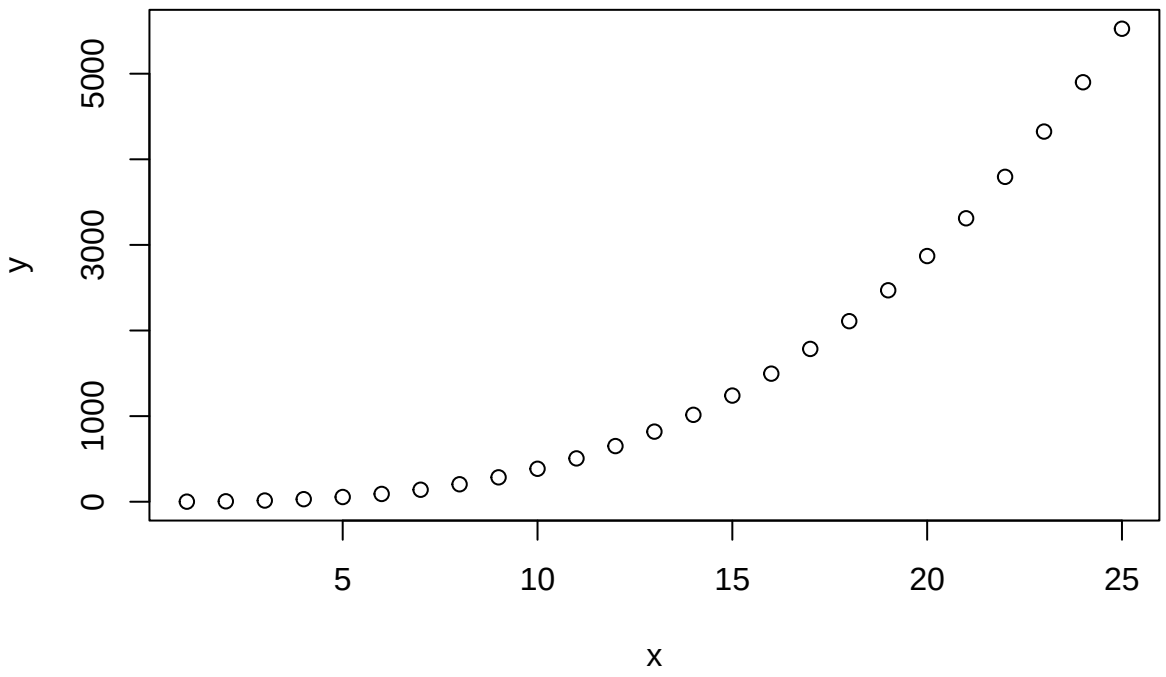
```r
s_n_2 <- sapply(1:25, compute_s_n)

s_n_2
```

```
##  [1]    1    5   14   30   55   91  140  204  285  385  506  650  819 1015 1240
## [16] 1496 1785 2109 2470 2870 3311 3795 4324 4900 5525
```

10. Repita o exercício 8, mas desta vez use `map_dbl`.

11. Crie o gráfico de S_n versus n, mas para $n = 1, \cdots, 25$.

```r
n <- 25
x <- 1:25
y<- sapply(x, compute_s_n)
plot(x, y)
```



12. Confirme a fórmula para esta soma é

$$S_n = \frac{n(n+1)(2n+1)}{6}$$

```r
square_sum <- function(n) {
  n * (n+1) * (2*n + 1) / 6
```

```
}


N <- 1:25
COMPUTE_S_N <- sapply(N, compute_s_n)
SQUARE_SUM <- sapply(N, square_sum)
IS_EQUAL <- identical(COMPUTE_S_N, SQUARE_SUM)

df <- data.frame(N, COMPUTE_S_N, SQUARE_SUM, IS_EQUAL)

df
```

```
##     N COMPUTE_S_N SQUARE_SUM IS_EQUAL
## 1   1           1          1     TRUE
## 2   2           5          5     TRUE
## 3   3          14         14     TRUE
## 4   4          30         30     TRUE
## 5   5          55         55     TRUE
## 6   6          91         91     TRUE
## 7   7         140        140     TRUE
## 8   8         204        204     TRUE
## 9   9         285        285     TRUE
## 10 10         385        385     TRUE
## 11 11         506        506     TRUE
## 12 12         650        650     TRUE
## 13 13         819        819     TRUE
## 14 14        1015       1015     TRUE
## 15 15        1240       1240     TRUE
## 16 16        1496       1496     TRUE
## 17 17        1785       1785     TRUE
## 18 18        2109       2109     TRUE
## 19 19        2470       2470     TRUE
## 20 20        2870       2870     TRUE
## 21 21        3311       3311     TRUE
## 22 22        3795       3795     TRUE
## 23 23        4324       4324     TRUE
## 24 24        4900       4900     TRUE
## 25 25        5525       5525     TRUE
```