ME115 - Linguagem R

Atividade Prática 03 - Gabarito

1º semestre de 2023

Introdução

Nessa atividade, exploraremos:

1. funções do tipo pré-definidas (*built-in*);
2. funções definidas pelo usuário: criar e executar funções.

Parte dos exercícios são do Capítulo 3 do Livro: https://rafalab.github.io/dsbook/programming-basics.html.

Atividade

1. A função `nchar()` retorna o total de caracteres de cada elemento de um vector da classe `character`. Considere o conjunto de dados `murders` do pacote `dslabs`. Escreva uma linha de código que aloca à variável `nome_novo` a sigla do estado quando o nome do mesmo tem mais do que 8 caracteres.

   Solução:

```
library(dslabs)
data(murders)

nome_novo <- ifelse(nchar(murders$state) > 8, murders$abb, murders$state)

head(cbind(murders[, 1:2],
           nchar = nchar(murders$state), nome_novo), 10)
```

```
##                   state abb nchar nome_novo
## 1               Alabama  AL     7   Alabama
## 2                Alaska  AK     6    Alaska
## 3               Arizona  AZ     7   Arizona
## 4              Arkansas  AR     8  Arkansas
## 5            California  CA    10        CA
## 6              Colorado  CO     8  Colorado
## 7           Connecticut  CT    11        CT
## 8              Delaware  DE     8  Delaware
## 9  District of Columbia  DC    20        DC
## 10              Florida  FL     7   Florida
```

2. Crie uma função chamada `soma_n()` que retorna, para um dado valor de `n`, a soma dos inteiros de 1 até `n` (inclusive). Use a função para determinar a soma de inteiros de 1 a 5000. Use a função `sum()` para validar sua função.

   Solução:

```
soma_n <- function(n){
  ## CORPO DA FUNCAO
  soma <- 0
  for (i in 1:n){
```

```
    soma <- soma + i
  }
  return(soma)
}

soma_n(5000)
```

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

Ou

```
soma_n <- function(n){
  a <- 1
  soma <- 0
  while(a <= n){
    soma <- soma + a
    a <- a + 1
  }
  return(soma)
}

soma_n(5000)
```

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

Ou, como sabemos que $S_n = \sum_{i=1}^{n} i = n(n+1)/2$, podemos escrever uma função usando essa fórmula:

```
soma_n <- function(n){
  n*(n + 1)/2
}

soma_n(5000)
```

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

3. Escreva uma função chamada `calcule_sn()` que, para qualquer valor n, calcule a soma $S_n = 1^2 + 2^2 + \ldots + n^2$. Aplique a função para $n = 10$.

   Solução:

```
calcule_sn <- function(n){
  soma <- 0
  for (i in 1:n){
    soma <- soma + i^2
    }
  return(soma)
  }

calcule_sn(10)
```

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

Ou

```
calcule_sn <- function(n){
  a <- 1
  soma <- 0
  while(a <= n){
```

```
      soma <- soma + a^2
      a <- a + 1
  }
  return(soma)
  }

calcule_sn(10)
```

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

4. 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, ..., S_{25} no vetor, usando `for` *loop* e a função `calcule_sn()`.

   Solução:

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

for(i in 1:n){
  s_n[i] <- calcule_sn(i)
}

head(cbind(n=1:25, s_n))
```

```
##      n s_n
## [1,] 1   1
## [2,] 2   5
## [3,] 3  14
## [4,] 4  30
## [5,] 5  55
## [6,] 6  91
```

5. Confirme que o resultado de $S_n = 1^2 + 2^2 + \ldots + n^2$ é $n(n+1)(2n+1)/6$.

   Solução:

```
n <- 1:25
S_n <- n*(n + 1)*(2*n + 1)/6
data.frame(n=n, s_n = s_n, S_n = S_n)
```

```
##     n  s_n  S_n
## 1   1    1    1
## 2   2    5    5
## 3   3   14   14
## 4   4   30   30
## 5   5   55   55
## 6   6   91   91
## 7   7  140  140
## 8   8  204  204
## 9   9  285  285
## 10 10  385  385
## 11 11  506  506
## 12 12  650  650
## 13 13  819  819
## 14 14 1015 1015
## 15 15 1240 1240
## 16 16 1496 1496
```

```
## 17 17 1785 1785
## 18 18 2109 2109
## 19 19 2470 2470
## 20 20 2870 2870
## 21 21 3311 3311
## 22 22 3795 3795
## 23 23 4324 4324
## 24 24 4900 4900
## 25 25 5525 5525
```

6. Crie uma função chamada `dados_rolando` que apresenta a soma do lançamento de `n` dados honestos. **Dica:** use a função `sample()`.

   Solução:

```
dados_rolando <- function(n){
  dados <- sample(x = 1:6, size = n, replace = TRUE)
  sum(dados)
}

## Executando a função dados_rolando() para n = 10, 100 e 1000.
c("n=10"   = dados_rolando(10),
  "n=100"  = dados_rolando(100),
  "n=1000" = dados_rolando(1000))
```

```
##   n=10  n=100 n=1000
##     40    343   3571
```

7. Crie uma função chamada `f_to_k` que converte graus em Fahrenheit para Kelvin. Calcule as temperaturas dos pontos de fusão (0°C ou 32°F) e ebulição (100°C ou 212°F) da água em graus Kelvin. Lembrando que:

$$\frac{t_C}{5} = \frac{t_F - 32}{9} = \frac{t_k - 273}{5}.$$

   Solução:

```
f_to_k <- function(tempF){
    tempK <- (tempF - 32) * 5/9 + 273
    return(tempK)
    }

f_to_k(32)
```

```
## [1] 273
```

```
f_to_k(212)
```

```
## [1] 373
```

```
f_to_k(c(fusao = 32, ebolicao = 212))
```

```
##    fusao ebolicao
##      273      373
```

8. Crie uma função chamada `stats()` que, tendo um vetor como argumento, retorna os valores da média, variância e desvio padrão. Calcule a média, variância e desvio padrão usando as respectivas fórmulas e não as funções pré-definidas no R. Use as funções `mean()`, `var()` e `sd()` para validar sua função.

   Solução:

```r
stats <- function(x){
  n <- sum(!is.na(x))

  # Média
  soma <- 0
  for(xi in x){
    if(!is.na(xi)) soma <- soma + xi
    }
  m <- soma/n

  # Variância
  soma2 <- 0
  for(xi in x) {
    if(!is.na(xi)) soma2 <- soma2 + (xi - m)^2
    }
  var <- soma2/(n - 1)

  sd <- sqrt(var)

  return(c(`Média` = m, `Variância` = var, `Desvio Padrão` = sd))
  }

x <- c(rep(1:5, each = 2), NA)
stats(x)
```

```
##         Média     Variância Desvio Padrão
##      3.000000      2.222222      1.490712
```

```r
c(mean(x, na.rm = TRUE), var(x, na.rm = TRUE), sd(x, na.rm = TRUE))
```

```
## [1] 3.000000 2.222222 1.490712
```

9. Utilize a função `runif()` para criar uma função que retorne um número aleatório inteiro entre 0 e 100 (0 e 100 inclusive).

   Solução:

```r
runif.Int <- function(n){
  amostra <- round(runif(n,0,100),0)
  return(amostra)
}

set.seed(2023)
runif.Int(1)
```

```
## [1] 47
```

10. Crie uma função chamada `multiplos3` com o intuito de receber como argumento um número natural n e retorna um array com os n primeiros múltiplos de 3. Aplique a função para $n = 20$.

    Solução:

```r
multiplos3 <- function(n){
  multiplos <- NULL
  for(i in 1:n){
    multiplos[i] <- 3*i
  }
  return(multiplos)
```

```
}

multiplos3(20)
```

```
##  [1]  3  6  9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60
```

Desafio

11. Escreva uma função chamada `fibonacci()` que retorna a sequência de fibonacci de um número n. Use a função para $n = 0$, $n = 1$, $n = 2$ e $n = 10$. Lembre que `fibonacci(0)` deve retornar 0 e `fibonati(1)` deve retornar 1.

    Solução:

```
fibonacci <- function(n){
  f <- vector("numeric", n + 1)
  f[1] <- 0
  f[2] <- 1

  if(n == 0){
    return(f[1])
  } else if(n == 1){
      return(f[1:2])
  } else {
      for(i in 3:(n + 1)){
        f[i] <- f[i - 1] + f[i - 2]
      }
      return(f)
  }
}
```

    Vamos testar alguns valores:

```
fibonacci(0)
```

```
## [1] 0
```

```
fibonacci(1)
```

```
## [1] 0 1
```

```
fibonacci(10)
```

```
##  [1]  0  1  1  2  3  5  8 13 21 34 55
```

12. Escreva uma função chamada `palindroma()` que recebe uma frase (palavra) e retorna "É palíndromo" se a frase é palíndroma e "Não é :(" caso contrário. Uma frase é palíndroma se ao ser lida da esquerda para a direita produz o mesmo resultado se lida da direita para a esquerda. Exemplo de palíndromos: "A torre da derrota", "ama", "Assim a aia ia a missa". Teste a função usando as frases exemplo.

    Solução:

```
## Código escrito pelo PED Rafael Branco

      palindroma <- function(palavra){

          palavra <- tolower(gsub(" ", "", palavra, fixed = T))
          n <- nchar(palavra)
```

```
          for (i in 1:floor(n/2)) {
              if (substring(palavra, i, i) != substring(palavra, n-(i-1), n-(i-1)))
                return("Não é :(")
          }

        return("É palíndromo")

        }

      palindroma("Assim a aia ia a missa")
```

```
## [1] "É palíndromo"
```

```
      palindroma("Assim i aia ia a missa")
```

```
## [1] "Não é :("
```

```
      palindroma("ama")
```

```
## [1] "É palíndromo"
```

```
      palindroma("A torre da derrota")
```

```
## [1] "É palíndromo"
```

Ou

```
palindroma <- function(frase){

  ## Essa função testa se uma frase/palavra é palíndroma
  frase <- tolower(gsub(" ", "", frase, fixed = TRUE))

  ## Escrever a frase em ordem inversa
  rev_frase <- paste(rev(strsplit(frase, split = "")[[1]]), collapse = "")

  ## Comparar a frase com a frase invertida
  ifelse(frase == rev_frase, "É palíndromo", "Não é :(")

}

palindroma("Assim a aia ia a missa")
```

```
## [1] "É palíndromo"
```

```
palindroma("Assim i aia ia a missa")
```

```
## [1] "Não é :("
```

```
palindroma("ama")
```

```
## [1] "É palíndromo"
```

```
palindroma("A torre da derrota")
```

```
## [1] "É palíndromo"
```

Agradecimentos

O material foi produzido pela Profa. Tatiana Benaglia para o curso de ME115 e alterado pelos professores Rafael e Larissa em 1S2023.