Introdução à Análise de Dados – Complementar para a G1 – 2023.1

1) Manipulação de vetores.

a) Defina um vetor **v** com a sequência de números ímpares maiores que zero e menores que 50.

```
v <- seq(1, 49, 2)
v
```

b) Calcule a soma dos elementos desse vetor.

```
soma <- sum(v)
soma
```

c) Selecione os elementos deste vetor que são múltiplos de 3 e os armazene em uma variável chamada **v.selecionado**.

```
v.selecionado <- v[v%%3==0]
v.selecionado
```

d) Apresente um vetor chamado **v.dividido** que contém a divisão por elemento da sequência obtida no item a) por 3.

```
v.dividido <- v / 3
v.dividido
```

2) Utilizando apenas as funções **c()**, **seq()** e **indexação de vetores []**, crie os seguintes objetos:

a) Uma sequência de vinte valores em intervalos regulares, indo de 0 a 100, nomeada **sq1**;

```
sq1 <- seq(0, 100, by = (100 - 0) / (20 - 1))
sq1
```

b) Um objeto, denominado **sq2**, que contenha todos os elementos de **sq1**, exceto o quinto e décimo quinto valores;

```
sq2 <- c(sq1[1:4], sq1[6:14], sq1[16:20])
sq2
```

c) Um vetor **sq3** contendo apenas as posições ímpares do objeto **sq1**;

```
sq3 <- sq1[seq(1, 19, 2)]
sq3
```

d) Uma sequência igual a **sq1** substituindo, apenas, os valores nas posições pares, pelo número relativo à sua posição. Denomine esse objeto de **sq4**.

```
sqaux <- 1:20
sqaux
sqaux1 <- sqaux %% 2
sqaux1
sqaux2 <- sqaux1 * (-1) + 1
sqaux2
sqaux2 <- sqaux * sqaux2
sqaux2
sq4 <- sq1 * sqaux1 + sqaux2
sq4
```

e) Utilizando o comando **matrix**, crie uma matriz 3 x 3 chamada **m** a partir dos dados armazenados em **sq1**.

```
m <- matrix(sq1, 3, 3)
m
```

f) Divida os elementos da matriz **m** por **2**.

```
m <- m / 2
m
```

3) Considere os vetores **u**, **v**, **w**, **x**, **y** e **z** e a matriz **m** formada a partir da união desses vetores, onde **u** será a primeira linha, **v**, a segunda e assim por diante:

u = (-1, 3, -8, 9, 10), v = (-8, 12, 4, 2, 3), w = (-2, -5, 3, 6, 7), x = (1, 5, 9, 12, 6), y = (2, 6, 10, -3, 8), z = (3, 7, 11, -10, 2)

Para os vetores **u**, **v**, **w**, **x**, **y** e **z** e a matriz **m**, escreva comandos em R para retornar:

```
u <- c(-1, 3,-8, 9, 10)
v <- c(-8, 12, 4, 2, 3)
w <- c(-2,-5, 3, 6, 7)
x <- c(1, 5, 9, 12, 6)
y <- c(2, 6, 10,-3, 8)
z <- c(3, 7, 11,-10, 2)
```

a) A média dos valores de x + y - u.

```
media <- mean (x + y - u)
media
```

b) Um vetor chamado **maior.igual** que identifica como **TRUE** os elementos do vetor resultante de **x + y + z** cujos valores são maiores ou iguais a **3** e como **FALSE** os demais elementos.

```
maior.igual = (x + y + z) >= 3
maior.igual
```

c) A matriz **m** criada a partir dos vetores.

```
m <- matrix(c(u, v, w, x, y, z), ncol = 5, byrow = TRUE)
m

m <- t(matrix(c(u, v, w, x, y, z), nrow = 5, ncol = 6))
m
```

d) O elemento que está na linha **3** e coluna **2** da matriz **m**.

```
elemento <- m[3, 2]
elemento
```

e) Um vetor contendo a segunda coluna da matriz **m**.

```
v <- m[, 2]
v
```

f) Uma matriz chamada **matriz.selecionada** contendo a primeira e a terceira linhas da matriz **m**.

```
matriz.selecionada <- m[c(1, 3), ]
matriz.selecionada
```

4) No estudo de escalas de temperatura, aprendemos que a conversão de graus Fahrenheit (F) para graus Celsius (C) se dá pela fórmula F= (9*C/5) + 32.

Elabore uma função em R chamada converte.temperatura que receba como parâmetro a temperatura que deve ser convertida e a escala (F para Fahrenheit ou C para Celsius) na qual a mesma se encontra e retorne a temperatura convertida, caso receba uma escala diferente de “F” ou “C” retorna **NaN**.

```
converte.temperatura <- function (temperatura, escala)
{
  resultado <- 0.0

  if (escala == "C")
    resultado <- (9 * temperatura / 5) + 32
  else if (escala == "F")
    resultado <- (5 * (temperatura - 32)) / 9
  else
    resultado <- NaN

  return(resultado)
}

print(converte.temperatura(32, "F"))
print(converte.temperatura(25, "C"))
print(converte.temperatura(30, "B"))
```

5) Elabore uma função chamada **conta.primos** que recebe os parâmetros **inicio** e **fim** e retorna o total de números primos no intervalo entre entres. Observe que **inicio** e **fim** também devem ser considerados.

```
conta.primos <- function (inicio, fim)
{
  d <- 2L
  primo <- FALSE
  contador <- 0

  for (i in inicio:fim)
  {
    d = 2L
    if (i <= 1)
    {
      primo <- FALSE
    }
    else
    {
      primo <- TRUE
      while ((primo == TRUE) & (d <= (i / 2)))
      {
        if ((i %% d) == 0)
        {
          primo <- FALSE
        }
        d <- d + 1
      }
    }
    if (primo == TRUE)
    {
      contador = contador + 1
    }
  }

  return (contador)

}

print(conta.primos (3, 10))
print(conta.primos (10, 20))
print(conta.primos (13, 40))
```

6) Usando as instruções **for** e **if** e as funções **nrow()** e **ncol()**, construa uma função chamada **calcula.soma** que recebe uma matriz **m** e calcula a soma dos elementos da matriz que possuem valor superior a **5**.

```
calcula.soma <- function (m)
{
  soma <- 0
  for (i in 1:nrow(m)) {
    for (j in 1:ncol(m)) {
      if (m [i, j] > 5) {
        soma <- soma + m [i, j]
      }
    }
  }
  return (soma)
}

m <- matrix (1:12, ncol = 4, nrow = 3)
print(calcula.soma (m))

m <- matrix (1:30, ncol = 6)
print(calcula.soma (m))
```

7) Manipulação de matriz.

a) Crie um objeto da classe **matriz** chamado **matriz.normal** com **30** linhas e **50** colunas contendo uma amostra de uma distribuição normal de média **10** e desvio padrão **3,6**.

```
matriz.normal <- matrix(rnorm(1500, mean=10, sd=3.6), nrow=30, ncol= 50)
matriz.normal
```

b) Apresente a **linha 2** e a seguir a **coluna 3** da matriz.

```
linha <- matriz.normal[2, ]
linha
coluna <- matriz.normal[, 3]
coluna
```

c) Apresente uma matriz chamada **matriz.selecionada** com os elementos da **matriz.normal** que estão nas posições definidas pelas linhas de **3** a **5** e pelas colunas de **8** a **10**.

```
matriz.selecionada <- matriz.normal[c(3, 4, 5), c(8, 9, 10)]
matriz.selecionada
```

d) Apresente as dimensões da matriz.

```
dim(matriz.normal)
```

e) Calcule a soma dos elementos da matriz.

```
soma <- sum(matriz.normal)
soma
```

f) Calcule o produto da matriz por sua matriz transposta.

```
produto <- matriz.normal %*% t(matriz.normal)
produto
```

g) Calcule a soma dos elementos da primeira linha da matriz.

```
soma <- sum(matriz.normal[1, ])
soma
```

h) Calcule média e variância por linha. Guarde os resultados em um data frame chamado **dataframe.linha**, cujas colunas são a média e a variância por linha e tem o nome de **media** e **variancia**, respectivamente. Faça o mesmo para as colunas, com o data frame de nome **dataframe.coluna** e os nomes das colunas **media** e **variancia**.

```
media <- apply(matriz.normal, 1, mean)
variancia <- apply(matriz.normal, 1, var)
dataframe.linha <- data.frame(media, variancia)
dataframe.linha

media <- apply(matriz.normal, 2, mean)
variancia <- apply(matriz.normal, 2, var)
dataframe.coluna <- data.frame(media, variancia)
dataframe.coluna
```

8) Manipulação de arrays.

a) Crie um *array* tri-dimensional com as dimensões **4**, **5** e **3** e o preencha com os sessenta primeiros números inteiros.

```
arraytridimensional <- array(0:59, c(4, 5, 3))
arraytridimensional
```

b) Calcule a soma dos elementos do *array* cuja coordenada na segunda dimensão é **4**.

```
soma <- sum(arraytridimensional[, 4, ])
soma
```

c) Calcule a média dos elementos do *array* cuja coordenada nas primeiras duas dimensões é **1**.

```
media <- mean(arraytridimensional[1, 1, ])
media
```

d) Calcule o *array* que se obtém multiplicando todos os valores do *array* por **2** e somando **5**.

```
arraynovo <- arraytridimensional * 2 + 5
arraynovo
```

9) Escreva uma função chamada **gera.fibonacci** que retorna um vetor com a sequência Fibonacci dado um parâmetro **n** de entrada que determina o número de elementos da sequência a serem retornados. Use o comando **for** na implementação.

```
# n deve ser maior 0
gera.fibonacci = function(n)
{
  # vetor para armazenar resultados
  fib = numeric(n)

  # condições iniciais
  fib[1] = 0
  if (n == 1) {
    return(fib)
  }

  fib[2] = 1
  if (n == 2) {
    return(fib)
  }

  # calculandos o números de 3 a n
  for (i in 3:n) {
    fib[i] = fib[i - 1] + fib[i - 2]
  }

  return(fib)
}

valor.n <- 10

gera.fibonacci(valor.n)
```

10) Usando a instrução **for** e a função **length()**, construa um procedimento chamado **apresenta.palavra** que recebe um vetor de caracteres (**vetor.caracteres**) e um vetor numérico (**vetor.numerico**). O vetor numérico armazena os índices a serem utilizados pelo procedimento para selecionar e apresentar os caracteres armazenados no vetor de caracteres.

```
apresenta.palavra <- function(vetor.caracteres, vetor.numerico)
{
  palavra <- ""

  for(i in 1:length(vetor.numerico))
  {
    print(vetor.caracteres[vetor.numerico[i]])
  }

}

vetor.caracteres <- c("A","B", "C", "D", "E", "F", "G", "H", "I", "J",
"K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X",
"Y", "Z")
vetor.numerico <- c(16, 21, 3, 18, 9, 15)

print(apresenta.palavra (vetor.caracteres, vetor.numerico))
```

11) Usando a instrução **for**, construa uma função chamada **transforma.vetor** que recebe um vetor de **n*m** posições, e gera como resultado uma matriz **n x m**. Observe que a matriz deve ser preenchida da esquerda para a direita e de cima para baixo, seguindo a ordem de posição dos elementos do vetor. Assim, o primeiro elemento deve estar na posição (1, 1), o segundo, na posição (1, 2) e assim por diante.

```
transforma.vetor = function (v, n, m)
{
  contador <- 0
  auxiliar <- rep(0, n * m)
  matriz <- matrix(auxiliar, nrow = n)

  for (i in 1:n)
  {
    for (j in 1:m)
    {
      contador <- contador + 1
      matriz [i, j] <- v [contador]
    }
  }

  return(matriz)
}

numero.linha <- 3
numero.coluna <- 4
vetor <- seq(1:(numero.linha*numero.coluna))

print(transforma.vetor(vetor, numero.linha, numero.coluna))
```

12) Usando as instruções **while** e **if** e a função **length()**, construa um procedimento chamado **encontra.valor** que recebe dois vetores numéricos **x** e **y** e apresenta na tela o total de elementos do primeiro vetor que também estão no segundo vetor. Considere também que: se um mesmo elemento de **x** aparecer mais de uma vez em **y**, deverá ser contado como uma única vez; se **x** apresentar elementos repetidos, eles deverão ser considerados como elementos distintos.

```
encontra.valor = function (x, y)
{
  total <- 0
  i <- 0
  while (i < length(x)) {
    i <- i + 1
    j <- 0
    while (j < length(y)) {
      j <- j + 1
      if (x [i] == y[j]) {
        total <- total + 1
        break
      }
    }
  }
  print (total)
}

x <- c (1, 5, 1, 100)
y <- c (0, 1, 2, 5, 3, 4, 5, 6, 7, 8, 9, 5)

encontra.valor (x, y)
```

13) Usando as instruções **for** e **if** e as funções **nrow()** e **ncol()**, construa uma função chamada **calcula.soma** que recebe uma matriz **m** e calcula a soma dos elementos da matriz que possuem valor superior a **5**.

```
calcula.soma = function (m)
{
  soma <- 0

  for (i in 1:nrow(m))
  {
    for (j in 1:ncol(m))
    {
      if (m [i, j] > 5)
      {
        soma <- soma + m [i, j]
      }
    }
  }

  return (soma)
}

m <- matrix (1:12, ncol = 4, nrow = 3)
print(calcula.soma (m))
```

14) Usando as instruções **for** e **if** e as funções **nrow()** e **ncol()**, construa uma função chamada **calcula.transposta** que recebe uma matriz **m** e calcula a sua matriz transposta. Verifique, usando funções prontas do R, se a função **calcula.transposta** está correta.

```
calcula.transposta <- function (m)
{
  v <- rep(0, ncol(m) * nrow(m))
  mt <- matrix(v, ncol = ncol(m))

  for (i in 1:nrow(m))
  {
    for (j in 1:ncol(m))
    {
      mt [j, i] <- m [i, j]
    }
  }

  return (mt)
}

v <- 1:9
m <- matrix(v, ncol = 3)

print(calcula.transposta(m))

print(t(m))
```

15) Usando as instruções **for** e **if** e as funções **nrow()** e **ncol()**, construa uma função chamada **calcula.valor** que recebe uma matriz **m** e calcula o quadrado da soma dos elementos da matriz que possuem valor superior a **3** e inferior a **6**.

```
calcula.valor <- function (m)
{
  soma <- 0

  for (i in 1:nrow(m)) {
    for (j in 1:ncol(m)) {
      if ((m [i, j] > 3) & (m [i, j] < 6)) {
        soma <- soma + m [i, j]
      }
    }
  }

  return (soma^2)
}

matriz <- matrix (1:12, ncol = 4)

print(calcula.valor (matriz))
```

16) Uma loja que vende motos deseja comparar vendas e preços de 2018 com as vendas e preços de 2010 de determinadas motos. As motos escolhidas, e seus respectivos preços e quantidades vendidas, estão na tabela a seguir.

	2010		2018	
Moto	**Preço (R$)**	**Quantidade**	**Preço (R$)**	**Quantidade**
Harley Iron	36.000,00	5	45.000,00	4
Honda Titan	12.000,00	10	13.000,00	15
Ninja MT	22.000,00	15	24.000,00	12

Para os dados acima:

a) Crie um data frame chamado **loja** contendo cinco colunas a saber: **Moto**, com os modelos das motos; **P2010**, com os preços das motos para 2010; **Q2010**, com a quantidade de motos vendidas em 2010; **P2018**, com os preços das motos para 2018; **Q2018**, com a quantidade de motos vendidas em 2018.

```
Moto <- c ("Harley Iron", "Honda Titan", "Ninja MT")
P2010 <- c (36000.00, 12000.00, 22000.00)
Q2010 <- c (5, 10, 15)
P2018 <- c (45000.00, 13000.00, 24000.00)
Q2018 <- c (4, 15, 12)

loja <- data.frame (Moto, P2010, Q2010, P2018, Q2018)
loja
```

b) Use uma função do R para apresentar a estrutura do data frame, e outra para apresentar o nome das colunas.

```
str (loja)
names (loja)
```

c) Usando funções do R, calcule para **2010** o menor valor para uma moto.

```
minimo <- sapply (loja["P2010"], min)
print(minimo[[1]])
```

d) Selecione no data frame apenas as colunas **P2018** e **Q2018** e respectivas linhas cuja quantidade vendida em **2010** seja superior a **5**. Devem ser utilizados mecanismos ou funções de filtro do R.

```
loja.f <- loja [loja$Q2010 > 5,]
loja.f <- loja.f [,  c("P2018", "Q2018")]
loja.f
```

e) Usando funções do R, crie uma coluna chamada **TotalVenda** no data frame que contenha, para cada uma das linhas do data frame, a soma das quantidades vendidas por moto em **2010** e **2018**.

```
loja$TotalVenda <- (loja$Q2010 + loja$Q2018)
loja
```