Lista 01 - MQ

Martha Gaudencio

2026-02-01

```
#MQ101 - Lista 01
#Nome: Martha Gaudencio da Silva
#Data: 07/10/2025
#Descrição: Exercícios sobre fundamentos do R (HOPR, Partes I, II)

#Exercício 1 - R como calculadora
10+2
```

```
## [1] 12
```

```
(10+2)*3
```

```
## [1] 36
```

```
((10+2)*3-6)/3
```

```
## [1] 10
```

```
#Os parênteses afetam o resultado porque na matemática existe uma sequência
#que deve ser seguida nas operações. Primeiro deve-se realizar as operações
#entre parênteses, depois se faz multiplicação ou divisão e por último adição
# e subtração. Isto influencia no cálculo de indicadores de políticas públicas,
# como taxas, que exigem correta multiplicação e divisão.

#Exercício 2 - Objetos e nomeação
x <-1:6
Name <- 1
name <- 0
Name + 1
```

```
## [1] 2
```

```
name +1
```

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

```
#Name gerou a variável com valor 1 e name gerou a variável com valor 0.
#As duas são diferentes porque o R diferencia letras maiúsculas de minúsculas.
# o "x" já refere-se a um intervalo de 1 a 6, considerando apenas números inteiros.

#Exercício 3 - Sorteio e reprodutibilidade
set.seed(123)
dice <- 1:6
sample(dice, size = 2, replace = TRUE)
```

```
## [1] 3 6
```

```
#Com o set.seed o resultado do sorteio foi o mesmo. Sem essa função, se dá
#a possibilidade de resultados diferentes. Assim, essa função gera
#reprodutibilidade dos resultados.

#Exercício 4 - Sua primeira função
roll2 <- function ( bones = 1:6) {
  dice <- sample ( bones , size = 2 , replace = TRUE )
  sum ( dice )
}
roll2 ()
```

```
## [1] 5
```

```
roll2 (1:20)
```

```
## [1] 29
```

```
#Foi criada uma função com o nome "roll2", "bones" é o argumento da função
#1:6 é para indicar os valores que integram a função. Como estamos simulando
#um dado, é para ser sorteado um número inteiro entre 1 e 6.
#dice é uma variável/um objeto novo, onde sample indica uma amostra aleatória,
#size = 2 indica que serão sorteados dois números e o "replace = TRUE"
#permite que haja repetição nos números sorteados. O sum(dice) realiza a soma
#dos itens sorteados. O roll2() considera o intervalo
#de 1 a 6, e o segundo roll2 considera de 1 a 20.

#Exercício 5 - ajuda e exemplos
?sample
```

```
## inicializando servidor httpd de ajuda ... concluído
```

```
example(sample)
```

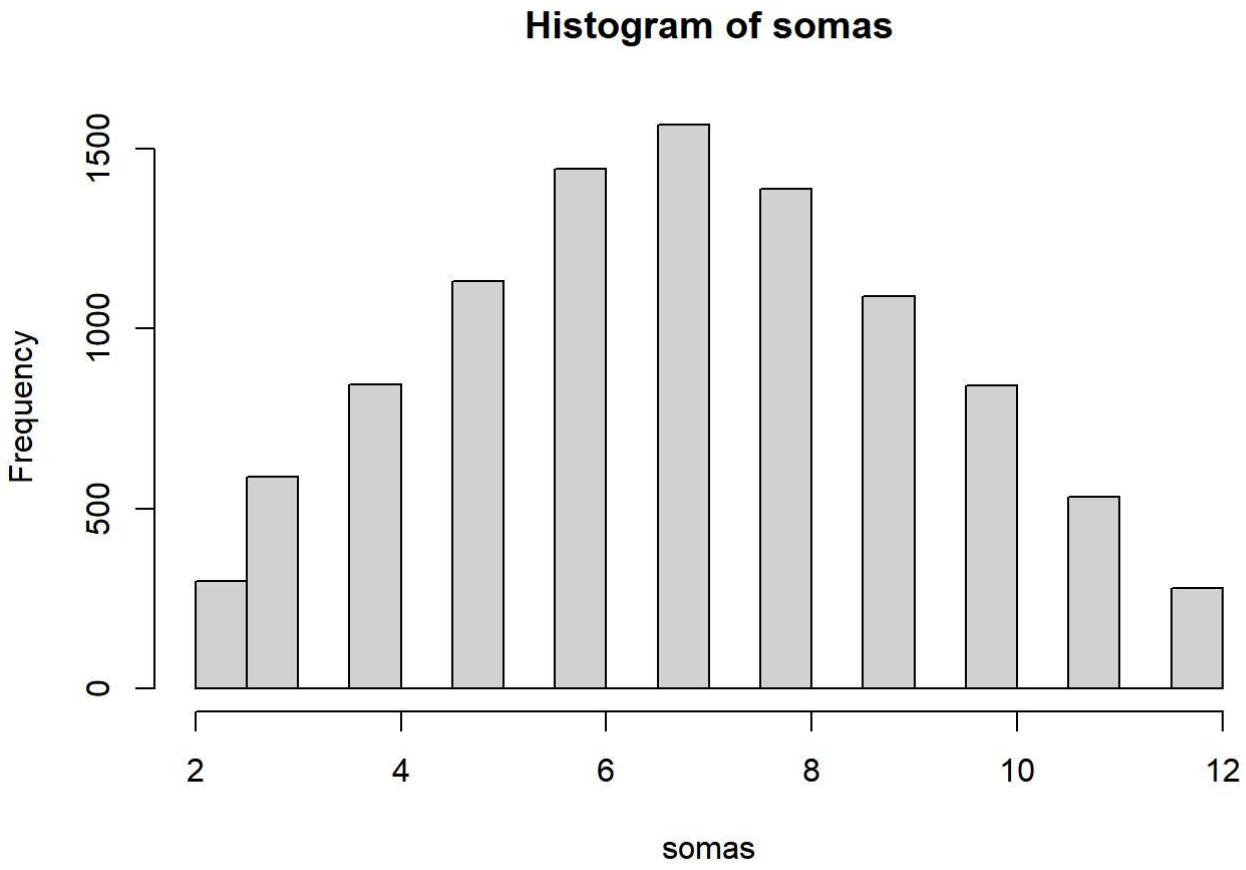
```
##
## sample> x <- 1:12
##
## sample> # a random permutation
## sample> sample(x)
##  [1] 11  9 12 10  5  3  2  6  1  8  4  7
##
## sample> # bootstrap resampling -- only if length(x) > 1 !
## sample> sample(x, replace = TRUE)
##  [1]  1 11  7  5 12 10  7  9  9 10  7 11
##
## sample> # 100 Bernoulli trials
## sample> sample(c(0,1), 100, replace = TRUE)
##   [1] 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 1 1 1 1 0 1 1 1 0
##  [38] 0 1 0 1 1 0 1 1 0 0 1 0 0 0 1 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 0 1 0 1 0 0
##  [75] 0 1 1 0 0 0 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1
##
## sample> ## More careful bootstrapping --  Consider this when using sample()
## sample> ## programmatically (i.e., in your function or simulation)!
## sample>
## sample> # sample()'s surprise -- example
## sample> x <- 1:10
##
## sample>     sample(x[x >  8]) # length 2
## [1]  9 10
##
## sample>     sample(x[x >  9]) # oops -- length 10!
##  [1]  8  1  7  3  5  2  6  4  9 10
##
## sample>     sample(x[x > 10]) # length 0
## integer(0)
##
## sample> ## safer version:
## sample> resample <- function(x, ...) x[sample.int(length(x), ...)]
##
## sample> resample(x[x >  8]) # length 2
## [1]  9 10
##
## sample> resample(x[x >  9]) # length 1
## [1] 10
##
## sample> resample(x[x > 10]) # length 0
## integer(0)
##
## sample> ## R 3.0.0 and later
## sample> sample.int(1e10, 12, replace = TRUE)
##  [1] 2705665808 3042067417 3922126562 9518754980 3525962555 8816911662
##  [7] 2406265538 4546118344 5032419856  952604626 2280966409 3975930314
##
## sample> sample.int(1e10, 12) # not that there is much chance of duplicates
##  [1] 2159599357  309471754 3157653709 8909399577  248931674 3781476631
##  [7] 8513198726 7641021114 2065442842 1523830150 5316106911 3666889664
```

```
#Para consultar a ajuda, se coloca uma interrogação antes do nome da função.
#O example(sample) retorna exemplos de códigos na ajuda da função sample.

#Exercício 6 - Simulação e histograma
set.seed (42)
somas <- replicate (10000,roll2 ())
length (somas)
```

```
## [1] 10000
```

```
hist (somas)
```




```
mean (somas); sd(somas)
```

```
## [1] 6.9532
```

```
## [1] 2.4329
```

```
#Exercício 7 - Tipos básicos
dbl <- c(1.5 , 2.0)
int <- c(1L,2L)
chr <- c("saude","educacao")
lgl <- c(TRUE , FALSE)
str (list ( dbl = dbl , int = int , chr = chr , lgl = lgl))
```

```
## List of 4
##  $ dbl: num [1:2] 1.5 2
##  $ int: int [1:2] 1 2
##  $ chr: chr [1:2] "saude" "educacao"
##  $ lgl: logi [1:2] TRUE FALSE
```

```
#O "str" mostra a estrutura de cada um dos dos valores. No caso, mostra a
#variável 'dbl' como numérica contínua, de 1,5 a 2; e 'int'como numérica apenas
#com números inteiros, no caso constando os números 1 e 2. "Chr" representa
#carácter de texto e 'lgl' relaciona-se aos vetores de verdadeiro ou falso.
#O uso prático dessa função é garantir o conhecimento da estrutura de cada
#elemento principalmente quando necessário para se realizar operações, por
#exemplo verificando se um número não está erroneamente armazenado como texto.

#Exercício 8 - data.frame

#Baralho
faces <- c("ace","two"," three "," four "," five ","six"," seven ",
           " eight "," nine ","ten"," jack "," queen "," king ")
suits <- c(" spades "," hearts "," diamonds "," clubs ")
deck <- data.frame (
  face = rep ( faces , times = 4) ,
  suit = rep ( suits , each = 13) ,
  value = rep (1:13 , times = 4)
)
nrow ( deck ) ; head ( deck , 10)
```

```
## [1] 52
```

```
##       face     suit value
## 1      ace  spades      1
## 2      two  spades      2
## 3   three   spades      3
## 4    four   spades      4
## 5    five   spades      5
## 6      six  spades      6
## 7   seven   spades      7
## 8   eight   spades      8
## 9    nine   spades      9
## 10     ten  spades     10
```

```
#O Baralho apresentado tem 3 colunas: "face", "suit" e "value" e 52 linhas,
#sendo que pela função "deck" foram exibidas no console apenas as 10 primeiras.

# Mini - base municipal
set.seed (2025)
municipios <- paste0 ("Mun_", sprintf ("%02d", 1:10))
dados_munic <- data.frame(
  municipio=municipios ,
  gasto_saude_pc = round ( runif (10 , 200 , 1200) , 2) ,
  taxa_evasao = round ( runif (10 , 0.00 , 0.20) , 3) ,
  taxa_desemprego = round ( rnorm (10 , 0.12 , 0.03) , 3)
)
head(dados_munic);summary (dados_munic)
```

```
##   municipio gasto_saude_pc taxa_evasao taxa_desemprego
## 1    Mun_01         932.62       0.087           0.108
## 2    Mun_02         675.76       0.190           0.067
## 3    Mun_03         714.22       0.131           0.107
## 4    Mun_04         698.43       0.005           0.143
## 5    Mun_05         980.28       0.094           0.152
## 6    Mun_06         704.25       0.171           0.114
```

```
##   municipio         gasto_saude_pc    taxa_evasao     taxa_desemprego
##  Length:10          Min.   : 327.9   Min.   :0.0050   Min.   :0.0670
##  Class :character   1st Qu.: 699.9   1st Qu.:0.0730   1st Qu.:0.1072
##  Mode  :character   Median : 741.9   Median :0.0905   Median :0.1160
##                     Mean   : 774.6   Mean   :0.0998   Mean   :0.1216
##                     3rd Qu.: 910.6   3rd Qu.:0.1467   3rd Qu.:0.1405
##                     Max.   :1098.4   Max.   :0.1900   Max.   :0.1930
```

```
#A mini base municipal tem 10 linhas, visto que consta 1:10 e são quatro
#colunas com as variáveis de município, gasto saúde, taxa de evasão e taxa
#de desemprego. A função summary mostra a quantidade de linhas, que as
#variáveis estão como carácter e também os números associados a cada uma delas.
#No caso: valores mínimo e máximo, mediana, média, 1º e 3º quartil de cada uma.
#Por exemplo, no exemplo hipotético a média de taxa de evasão seria de 9,98%.

#Exercício 9 - Seleção e filtros

deck [1 , ]
```

```
##   face     suit value
## 1  ace  spades      1
```

```
deck [c(1,3,5) , c("face","suit") ]
```

```
##      face     suit
## 1     ace  spades
## 3   three   spades
## 5    five   spades
```

```
deck [ -(1:48), ]
```

```
##       face    suit value
## 49     ten  clubs     10
## 50   jack   clubs     11
## 51  queen   clubs     12
## 52   king   clubs     13
```

```
subset_hearts <- deck [ deck $ suit == "hearts", ]
nrow (subset_hearts)
```

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

```
evaz_alta <- dados_munic [ dados_munic $ taxa_evasao > 0.10 , ]
evaz_alta
```

```
##   municipio gasto_saude_pc taxa_evasao taxa_desemprego
## 2    Mun_02         675.76       0.190           0.067
## 3    Mun_03         714.22       0.131           0.107
## 6    Mun_06         704.25       0.171           0.114
## 9    Mun_09         844.67       0.152           0.081
```

```
nrow(evaz_alta)
```

```
## [1] 4
```

```
#Na parte do baralho, o deck retorna a primeira linha e mostra todas as colunas,
#no caso sendo as colunas 'face', 'suit' e 'value'. Depois o deck mostra as
#linhas 1, 3 e 5 e apenas as colunas 'face' e 'suit'. Depois, com o sinal de
#negativo se excluíram as linhas de 1 a 48, mostrando apenas as últimas 4, no
#caso de 49 a 52. O último comando buscava pela carta 'hearts' mas ela foi
#criada dentro da função 'faces' com espaços e por isso não foi identificada,
#gerando o zero.

#Quanto aos municípios, foram apresentados quatro (o 2, 3, 6 e 9) com taxa de
#de evasão acima de 0.10, após o filtro selecionado.

#Exercício 10 - Modificando valores e NA

deck2 <- deck
deck2 $ value [c(13 , 26 , 39 , 52) ] <- 14
head ( deck2 , 13)
```

```
##       face     suit value
## 1      ace   spades      1
## 2      two   spades      2
## 3    three   spades      3
## 4     four   spades      4
## 5     five   spades      5
## 6      six   spades      6
## 7    seven   spades      7
## 8    eight   spades      8
## 9     nine   spades      9
## 10     ten   spades     10
## 11    jack   spades     11
## 12   queen   spades     12
## 13    king   spades     14
```

```
vals <- c( NA , 1:5)
mean ( vals )
```

```
## [1] NA
```

```
mean ( vals , na.rm = TRUE )
```

```
## [1] 3
```

```
dados_m2 <- dados_munic
dados_m2$ taxa_evasao [3] <- NA
dados_m2$ gasto_saude_pc [7] <- NA
mean ( dados_m2$ taxa_evasao )
```

```
## [1] NA
```

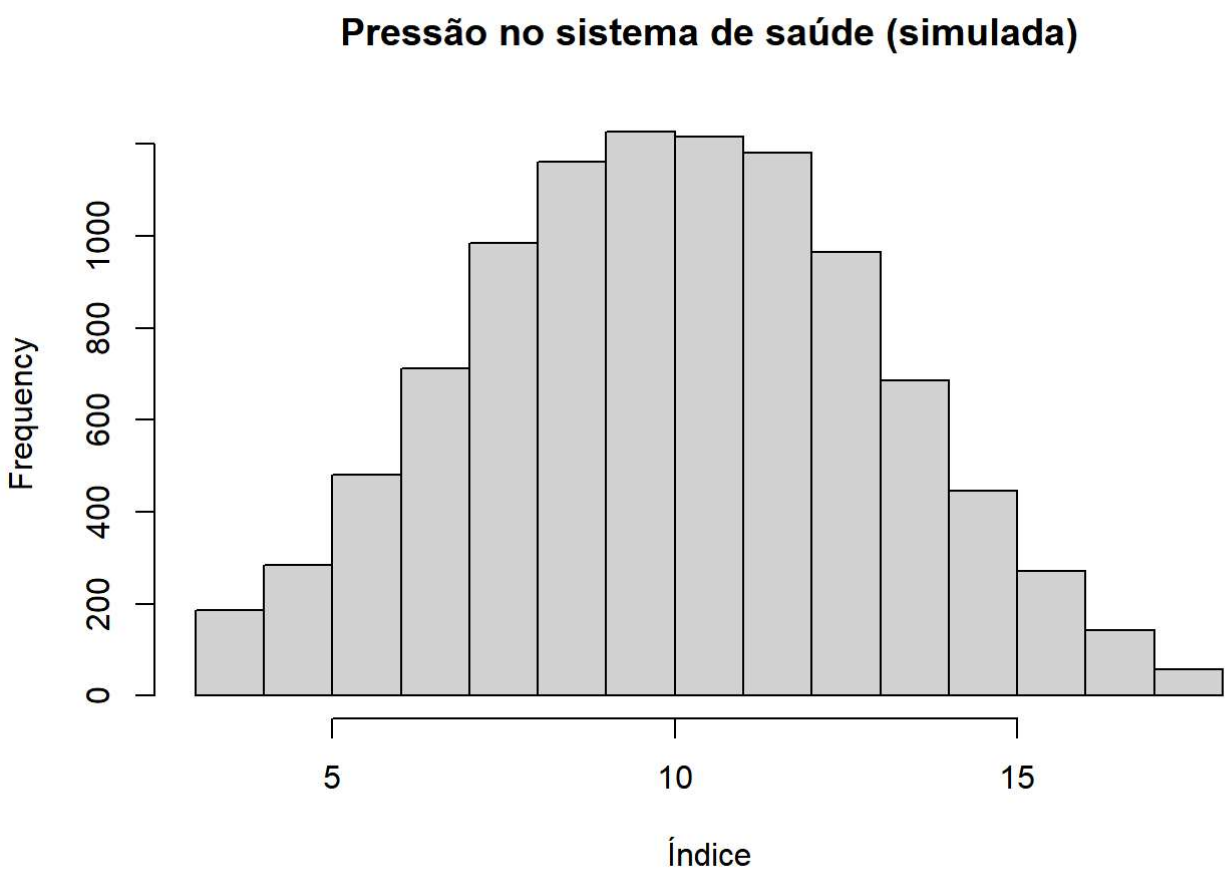
```
mean ( dados_m2$ taxa_evasao , na.rm = TRUE )
```

```
## [1] 0.09633333
```

```
# O uso de "na.rm = TRUE" é para ignorar o NA para permitir a realização de
# cálculos, como a média.

#Exercício 12 - Mini-projeto integrador: Saúde

set.seed (123)
pressao_saude <- function () {
  demanda <- sample (1:6 , 1 , TRUE )
  equipe <- sample (1:6 , 1 , TRUE )
  insumos <- sample (1:6 , 1 , TRUE )
  demanda + equipe + insumos
}
prs <- replicate(10000, pressao_saude())
hist(prs, main = "Pressão no sistema de saúde (simulada)", xlab = "Índice")
```



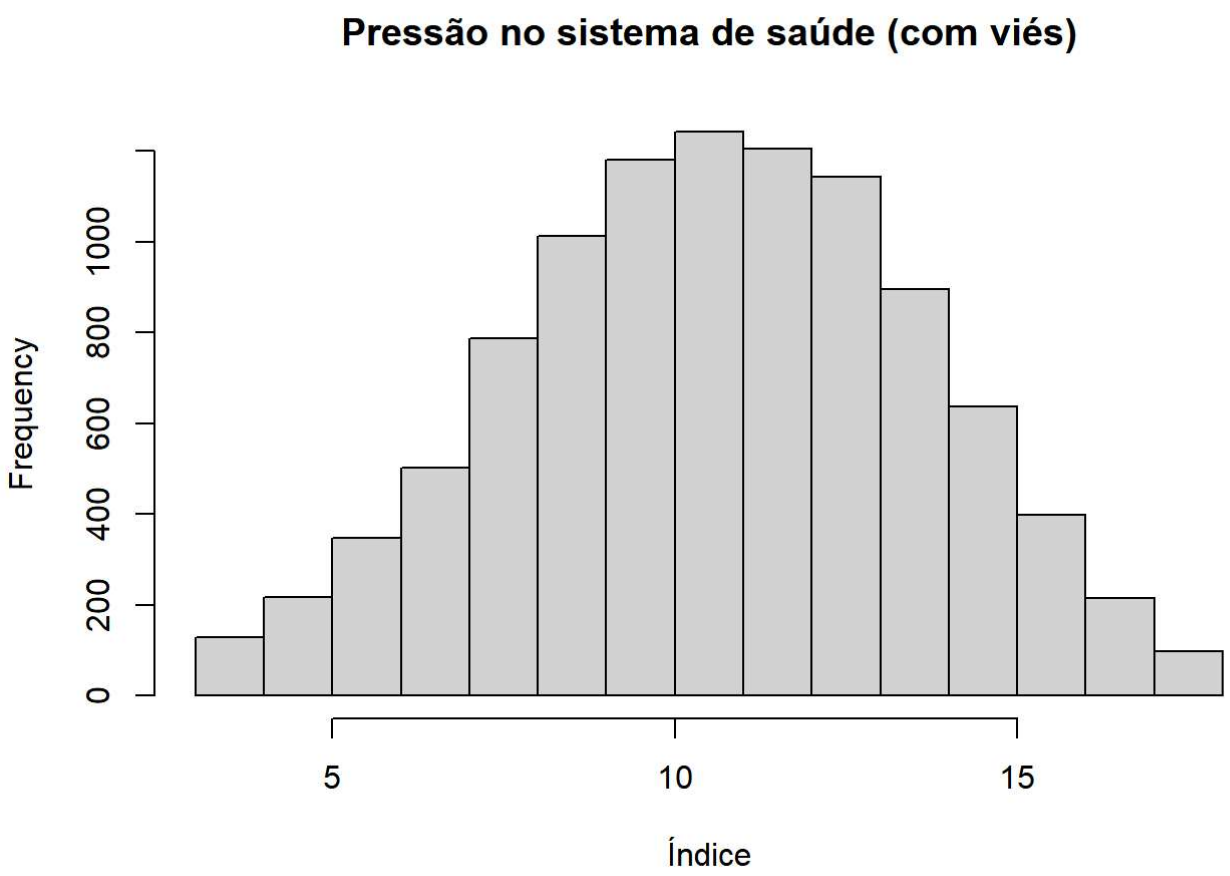

```
mean(prs); sd(prs)
```

```
## [1] 10.4759
```

```
## [1] 2.97498
```

```
# Viés em demanda (favorece 6)

prob_demanda <- c(rep(1/8, 5), 3/8)
pressao_vies <- function() {
  demanda <- sample(1:6, 1, TRUE, prob = prob_demanda)
  equipe  <- sample(1:6, 1, TRUE)
  insumos <- sample(1:6, 1, TRUE)
  demanda + equipe + insumos
}
prs_bias <- replicate(10000, pressao_vies())
hist(prs_bias, main = "Pressão no sistema de saúde (com viés)", xlab = "Índice")
```




```
mean(prs_bias) - mean(prs)
```

```
## [1] 0.6377
```

```
#A diferença de médias pelo cálculo mean(prs_bias) - mean(prs) mostra que é
#0,6377 entre as variáveis de pressão de saúde com e sem viés. O viés se deu
#pela inclusão de 'prob = prob_demanda', a partir da qual se deu 1/8 de chances
#de aparecer e 6 tem 3/8, aparecendo mais. Isso gera uma alteração também no
#histograma que ficou com uma 'calda' deslocada para a direita com viés,
#diferente do primeiro em formato de sino.
```