

Capítulo 02: Noções básicas de R

Igo da Costa Andrade

2023-10-31

2.11 Exercícios

1. Qual é a soma dos primeiros 100 números inteiros positivos? A fórmula para a soma dos inteiros de 1 até n é $n(n+1)/2$. Defina $n = 100$ e então use R para calcular a soma de 1 até 100 usando a fórmula. Qual é a soma?

```
n <- 100  
  
soma = n * (n+1) / 2
```

Resposta: A soma dos primeiros 100 inteiros positivos é 5050.

2. Agora use a mesma fórmula para calcular a soma dos inteiros de 1 a 1000.

```
n <- 1000  
  
soma = n * (n+1) / 2
```

Resposta: A soma dos primeiros 1000 inteiros positivos é 500500.

3. Observe o resultado da digitação do seguinte código em R:

```
n <- 1000  
x <- seq(1, n)  
sum(x)
```

```
## [1] 500500
```

Com base no resultado, o que você acha que as funções `seq` e `sum` fazem?

- a. `sum` cria uma lista de números e `seq` os soma.
 - b. `seq` cria uma lista de números e `sum` os soma.
 - c. `seq` cria uma lista aleatória e `sum` calcula a soma de 1 a 1.000.
 - d. `sum` sempre retorna o mesmo número.
4. Em matemática e programação, dizemos que avaliamos uma função quando substituímos o argumento por um determinado número. Então, se digitarmos `sqrt(4)`, avaliaremos a função `sqrt`. Em R, você pode avaliar uma função dentro de outra função. As avaliações acontecem de dentro para fora. Use uma linha de código para calcular o logaritmo, na base 10, da raiz quadrada de 100.

```
log(sqrt(100), base=10)
```

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

5. Qual das opções a seguir sempre retornará o valor numérico armazenado em `x`?

- a. `log(10^x)`
- b. `log10(x^10)`

c. `log(exp(x))`

d. `log(x, base=2)`

6. Certifique-se de que o conjunto de dados de assassinatos nos EUA esteja carregado. Use a função `str` para examinar a estrutura do objeto `murders`. Qual das alternativas a seguir descreve melhor as variáveis representadas neste *data frame*.

a. Os 51 estados.

b. As taxas de homicídio em todos os 50 estados e DC.

c. O nome do estado, a abreviatura do nome do estado, a região do estado e a população do estado e o número total de assassinatos em 2010.

d. `str` não apresenta informações relevantes.

```
library(dslabs)
data("murders")
```

```
str(murders)
```

```
## 'data.frame':  51 obs. of  5 variables:
## $ state      : chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
## $ abb       : chr  "AL" "AK" "AZ" "AR" ...
## $ region    : Factor w/ 4 levels "Northeast","South",...: 2 4 4 2 4 4 1 2 2 2 ...
## $ population: num  4779736 710231 6392017 2915918 37253956 ...
## $ total     : num  135 19 232 93 1257 ...
```

7. Quais são os nomes das colunas usadas pelo *data frame* para essas cinco variáveis?

```
colnames(murders)
```

```
## [1] "state"      "abb"        "region"     "population" "total"
```

8. Use o acessador `$` para extrair as abreviações de estado e atribuí-las ao objeto `a`. Qual é a classe deste objeto?

```
a <- murders$abb
```

```
class(a)
```

```
## [1] "character"
```

9. Agora use os colchetes para extrair as abreviações de estado e atribuí-las ao objeto `b`. Use a função `identical` para determinar se `a` e `b` são iguais.

```
b <- murders[['abb']]
```

```
identical(a, b)
```

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

10. Vimos que a coluna `region` armazena um fator. Você pode corroborar isso digitando:

```
class(murders$region)
```

```
## [1] "factor"
```

Com uma linha de código, use as funções `levels` e `length` para determinar o número de regiões definidas por este conjunto de dados.

```
length(levels(murders$region))
```

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

11. A função `table` pega um vetor e retorna a frequência de cada elemento. Você pode ver rapidamente quantos estados existem em cada região aplicando esta função. Use esta função em uma linha de código para criar uma tabela de estados por região.

```
table(murders$region)
```

```
##
##      Northeast      South North Central      West
##           9          17          12          13
```

12. Use a função `c` para criar um vetor com as altas temperaturas médias em janeiro para Pequim, Lagos, Paris, Rio de Janeiro, San Juan e Toronto, que são 35, 88, 42, 84, 81 e 30 graus Fahrenheit. Chame o objeto `temp`.

```
temp <- c(35, 88, 42, 84, 81, 30)
```

13. Agora crie um vetor com os nomes das cidades e chame o objeto `city`.

```
city <- c("Pequim", "Lagos", "Paris", "Rio de Janeiro", "San Juan", "Toronto")
```

14. Utilize a função `names` e os objetos definidos nos exercícios anteriores para associar os dados de temperatura à sua cidade correspondente.

```
names(temp) <- city
```

```
temp
```

```
##      Pequim      Lagos      Paris Rio de Janeiro      San Juan
##       35       88       42       84       81
##      Toronto
##       30
```

15. Utilize os operadores `[` e `:` para acessar a temperatura das três primeiras cidades da lista.

```
temp[1:3]
```

```
## Pequim Lagos Paris
##    35    88    42
```

16. Use o operador `[` para acessar a temperatura de Paris e San Juan.

```
temp[c("Paris", "San Juan")]
```

```
##      Paris San Juan
##       42      81
```

17. Use o operador `:` para criar a sequência de números 12, 13, 14, ..., 73.

```
vec <- seq(from=12, to=73)
```

```
vec
```

```
## [1] 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
## [26] 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
## [51] 62 63 64 65 66 67 68 69 70 71 72 73
```

18. Crie um vetor contendo todos os números ímpares positivos menores que 100.

```
impares_menores_que_100 <- seq(from=1, to=100, by=2)
```

```
impares_menores_que_100
```

```
## [1] 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
## [26] 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

19. Crie um vetor de números que comece em 6, não passe de 55 e adicione números em incrementos de $4/7$: 6, $6 + 4/7$, $6 + 8/7$ e assim por diante. Quantos números tem a lista? Dica: use `seq` e `length`.

```
vec <- seq(from=6, to=55, by=4/7)

length(vec)
```

```
## [1] 86
```

20. Qual é a classe do seguinte objeto `a <- seq(1, 10, 0.5)`?

```
a <- seq(1, 10, 0.5)

class(a)
```

```
## [1] "numeric"
```

21. Qual é a classe do seguinte objeto `a <- seq(1, 10)`?

```
a <- seq(1, 10)

class(a)
```

```
## [1] "integer"
```

22. A classe de `class(a<-1)` é numérica, não inteira. O padrão de R é numérico e para forçar um número inteiro, você precisa adicionar a letra L. Confirme se a classe de `1L` é inteira.

```
class(1)
```

```
## [1] "numeric"
```

```
class(1L)
```

```
## [1] "integer"
```

23. Defina o seguinte vetor:

```
x <- c("1", "3", "5")
```

e use coerção para obter números inteiros.

```
x <- as.numeric(x)

x
```

```
## [1] 1 3 5
```

24. Para os exercícios 24 a 31 usaremos o conjunto de dados de assassinatos nos EUA. Certifique-se de carregá-lo antes de começar. Use o operador `$` para acessar os dados do tamanho da população e armazená-los como objeto `pop`. Em seguida, use a função `sort` para redefinir `pop` para que seja classificado. Finalmente, use o operador `[` para relatar o menor tamanho da população.

```
library("dslabs")

data("murders")

pop <- murders$population
```

```
pop <- sort(pop)
```

```
pop[1]
```

```
## [1] 563626
```

25. Agora, em vez do menor tamanho populacional, encontre o índice da entrada com o menor tamanho populacional. Dica: use `order` em vez de `sort`.

```
indices <- order(murders$population)
```

```
i_min <- indices[1]
```

```
i_min
```

```
## [1] 51
```

26. Na verdade, podemos realizar a mesma operação do exercício anterior usando a função `which.min`. Escreva uma linha de código que faça isso.

```
i_min <- which.min(murders$population)
```

```
i_min
```

```
## [1] 51
```

27. Agora sabemos quão pequeno é o menor estado e qual linha o representa. Qual estado é esse? Defina uma variável `states` para ser os nomes dos estados do *data frame* `murders`. Informe o nome do estado com menor população.

```
states <- murders$state
```

```
states[i_min]
```

```
## [1] "Wyoming"
```

28. Você pode criar um *data frame* usando a função `data.frame`. Aqui está um exemplo rápido:

```
temp <- c(35, 88, 42, 84, 81, 30)
```

```
city <- c("Beijing", "Lagos", "Paris", "Rio de Janeiro", "San Juan", "Toronto")
```

```
city_temps <- data.frame(name=city, temperature=temp)
```

Use a função `rank` para determinar a classificação da população de cada estado, do menor ao maior tamanho populacional. Salve essas classificações em um objeto chamado `ranks` e crie um *data frame* com o nome do estado e sua classificação. Chame o quadro de dados `my_df`.

```
ranks <- rank(murders$population)
```

```
state <- murders$state
```

```
my_df <- data.frame(state, ranks)
```

```
my_df
```

```
##           state ranks
## 1      Alabama    29
## 2       Alaska     5
## 3      Arizona    36
## 4    Arkansas    20
```

## 5	California	51
## 6	Colorado	30
## 7	Connecticut	23
## 8	Delaware	7
## 9	District of Columbia	2
## 10	Florida	49
## 11	Georgia	44
## 12	Hawaii	12
## 13	Idaho	13
## 14	Illinois	47
## 15	Indiana	37
## 16	Iowa	22
## 17	Kansas	19
## 18	Kentucky	26
## 19	Louisiana	27
## 20	Maine	11
## 21	Maryland	33
## 22	Massachusetts	38
## 23	Michigan	43
## 24	Minnesota	31
## 25	Mississippi	21
## 26	Missouri	34
## 27	Montana	8
## 28	Nebraska	14
## 29	Nevada	17
## 30	New Hampshire	10
## 31	New Jersey	41
## 32	New Mexico	16
## 33	New York	48
## 34	North Carolina	42
## 35	North Dakota	4
## 36	Ohio	45
## 37	Oklahoma	24
## 38	Oregon	25
## 39	Pennsylvania	46
## 40	Rhode Island	9
## 41	South Carolina	28
## 42	South Dakota	6
## 43	Tennessee	35
## 44	Texas	50
## 45	Utah	18
## 46	Vermont	3
## 47	Virginia	40
## 48	Washington	39
## 49	West Virginia	15
## 50	Wisconsin	32
## 51	Wyoming	1

29. Repita o exercício anterior, mas desta vez ordene `my_df` de forma que os estados sejam ordenados do menos populoso para o mais populoso. Dica: crie um objeto `ind` que armazene os índices necessários para ordenar os valores da população. Em seguida, use o operador de colchetes `[` para reordenar cada coluna no quadro de dados.

```
ind <- order(murders$population)
```

```

state <- murders$state

state <- state[ind]
pop <- murders$population

pop <- pop[ind]

my_df <- data.frame(state, pop)

my_df

```

```

##           state      pop
## 1      Wyoming  563626
## 2 District of Columbia 601723
## 3      Vermont  625741
## 4    North Dakota  672591
## 5        Alaska  710231
## 6    South Dakota  814180
## 7      Delaware  897934
## 8      Montana  989415
## 9    Rhode Island 1052567
## 10   New Hampshire 1316470
## 11        Maine  1328361
## 12       Hawaii  1360301
## 13       Idaho  1567582
## 14       Nebraska 1826341
## 15   West Virginia 1852994
## 16     New Mexico 2059179
## 17       Nevada  2700551
## 18        Utah  2763885
## 19       Kansas  2853118
## 20     Arkansas  2915918
## 21   Mississippi 2967297
## 22        Iowa  3046355
## 23   Connecticut 3574097
## 24     Oklahoma  3751351
## 25       Oregon  3831074
## 26     Kentucky  4339367
## 27     Louisiana  4533372
## 28   South Carolina 4625364
## 29       Alabama  4779736
## 30       Colorado 5029196
## 31     Minnesota  5303925
## 32     Wisconsin  5686986
## 33       Maryland  5773552
## 34       Missouri  5988927
## 35     Tennessee  6346105
## 36       Arizona  6392017
## 37       Indiana  6483802
## 38   Massachusetts 6547629
## 39     Washington  6724540
## 40       Virginia  8001024
## 41     New Jersey  8791894
## 42   North Carolina 9535483

```

```
## 43          Michigan  9883640
## 44           Georgia  9920000
## 45            Ohio 11536504
## 46    Pennsylvania 12702379
## 47       Illinois 12830632
## 48        New York 19378102
## 49         Florida 19687653
## 50          Texas 25145561
## 51    California 37253956
```

30. O vetor `na_example` representa uma série de contagens. Você pode examinar rapidamente o objeto usando:

```
str(na_example)
```

```
## int [1:1000] 2 1 3 2 1 3 1 4 3 2 ...
```

No entanto, quando calculamos a média com a função `mean`, obtemos NA:

```
mean(na_example)
```

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

A função `is.na` retorna um vetor lógico que nos informa quais entradas são NA. Atribua este vetor lógico a um objeto chamado `ind` determine quantos NAs `na_example` possui.

```
ind <- is.na(na_example)
```

```
sum(ind)
```

```
## [1] 145
```

31. Agora calcule a média novamente, mas apenas para as entradas que não são NA. Dica: lembre-se do operador `!`.

```
mean(na_example[!ind])
```

```
## [1] 2.301754
```

32. Anteriormente criamos este quadro de dados:

```
temp <- c(35, 88, 42, 84, 81, 30)
city <- c("Beijing", "Lagos", "Paris", "Rio de Janeiro", "San Juan", "Toronto")
city_temps <- data.frame(name = city, temperature = temp)
```

Refaça o quadro de dados usando o código acima, mas adicione uma linha que converta a temperatura de Fahrenheit para Celsius. A conversão é $C = \frac{5}{9} \times (F - 32)$.

```
temp_f <- c(35, 88, 42, 84, 81, 30)
```

```
temp_c <- round((5/9) * (temp_f - 32), 2)
```

```
city <- c("Beijing", "Lagos", "Paris", "Rio de Janeiro", "San Juan", "Toronto")
city_temps <- data.frame(name = city, temp_Fahrenheit = temp_f, temp_Celsius = temp_c)
```

```
city_temps
```

```
##           name temp_Fahrenheit temp_Celsius
## 1    Beijing             35           1.67
## 2     Lagos             88           31.11
## 3     Paris             42           5.56
## 4 Rio de Janeiro         84          28.89
```



```
## 5      San Juan      81      27.22
## 6      Toronto     30      -1.11
```

33. Qual é a seguinte soma $1 + 1/2^2 + 1/3^2 + \dots + 1/100^2$? Dica: graças a Euler, sabemos que deveria estar próximo de $\pi^2/6$.

```
denominador <- seq(from=1, to=100)

x <- 1 / denominador ^2

sum(x)
```

```
## [1] 1.634984
```

34. Calcule a taxa de homicídios por 100.000 para cada estado e armazene-a no objeto `murder_rate`. Em seguida, calcule a taxa média de homicídios nos EUA usando a função `mean`. Qual é a média?

```
library(dslabs)
data("murders")

murders_rate <- murders$total / murders$population * 10^5

mean(murders_rate)
```

```
## [1] 2.779125
```

35. Para os exercícios restantes 35-42, comece carregando a biblioteca e os dados.

```
library(dslabs)
```

Calcule a taxa de homicídios por 100.000 para cada estado e armazene-a em um objeto chamado `murder_rate`. Em seguida, use operadores lógicos para criar um vetor lógico chamado `low` que nos diga quais entradas de `murder_rate` são menores que 1.

```
library(dslabs)

data(murders)

murder_rate <- murders$total / murders$population * 10^5

low <- murder_rate[murders_rate < 1]

low
```

```
## [1] 0.5145920 0.7655102 0.6893484 0.8280881 0.9992600 0.3798036 0.5947151
## [8] 0.9396843 0.9825837 0.7959810 0.3196211 0.8871131
```

36. Agora use os resultados do exercício anterior e a função `which` para determinar os índices `murder_rate` associados a valores inferiores a 1.

```
id_low <- which(murder_rate %in% low)
id_low
```

```
## [1] 12 13 16 20 24 30 35 38 42 45 46 51
```

37. Utilize os resultados do exercício anterior para reportar os nomes dos estados com taxas de homicídio inferiores a 1.

```
murders[id_low, ]
```

```
##           state abb      region population total
```

## 12	Hawaii	HI	West	1360301	7
## 13	Idaho	ID	West	1567582	12
## 16	Iowa	IA	North Central	3046355	21
## 20	Maine	ME	Northeast	1328361	11
## 24	Minnesota	MN	North Central	5303925	53
## 30	New Hampshire	NH	Northeast	1316470	5
## 35	North Dakota	ND	North Central	672591	4
## 38	Oregon	OR	West	3831074	36
## 42	South Dakota	SD	North Central	814180	8
## 45	Utah	UT	West	2763885	22
## 46	Vermont	VT	Northeast	625741	2
## 51	Wyoming	WY	West	563626	5

38. Agora estenda o código dos exercícios 2 e 3 para relatar os estados do Nordeste com taxas de homicídios inferiores a 1. Dica: use o vetor lógico definido anteriormente `low` e o operador lógico `&`.

```
rate <- murders$total / murders$population * 10^5

low_and_northeast <- murders$rate[murders$region=="Northeast" & murders$rate < 1]

id_low_and_northeast <- which(murders$rate %in% low_and_northeast)

murders[id_low_and_northeast, ]

## [1] state      abb      region      population total
## <0 linhas> (ou row.names de comprimento 0)
```

39. Num exercício anterior calculamos a taxa de homicídios para cada estado e a média destes números. Quantos estados estão abaixo da média?

```
library(dslabs)

data("murders")

murders$rate <- murders$total / murders$population * 10^5

mean_rate <- mean(murders$rate)

length(murders$rate[murders$rate < mean_rate])

## [1] 27
```

40. Use a função de correspondência para identificar os estados com as abreviações AK, MI e IA. Dica: comece definindo um índice das entradas `murders$abb` que correspondem às três abreviações e, em seguida, use o operador `[` para extrair os estados.

```
library(dslabs)

data("murders")

abbs <- c("AK", "MI", "IA")

ind <- match(abbs, murders$abb)

murders$state[ind]
```

```
## [1] "Alaska" "Michigan" "Iowa"
```

41. Use o operador `%in%` para criar um vetor lógico que responda à pergunta: quais das seguintes são abreviações reais: MA, ME, MI, MO, MU?| para criar um vetor lógico que responda à pergunta: quais das seguintes são abreviações reais: MA, ME, MI, MO, MU?| para criar um vetor lógico que responda à pergunta: quais das seguintes são abreviações reais: MA, ME, MI, MO, MU?

```
library(dslabs)

data("murders")

abbs <- c("MA", "ME", "MI", "MO", "MU")

abbs %in% murders$abb
```

```
## [1] TRUE TRUE TRUE TRUE FALSE
```

42. Estenda o código usado no exercício 7 para relatar a única entrada que não é uma abreviatura real. Dica: utilize o operador `!`, que se transforma `FALSE` em `TRUE` e vice-versa, `|verb|which|` para obter um índice.

```
library(dslabs)

data("murders")

abbs <- c("MA", "ME", "MI", "MO", "MU")

abbs[!(abbs %in% murders$abb)]
```

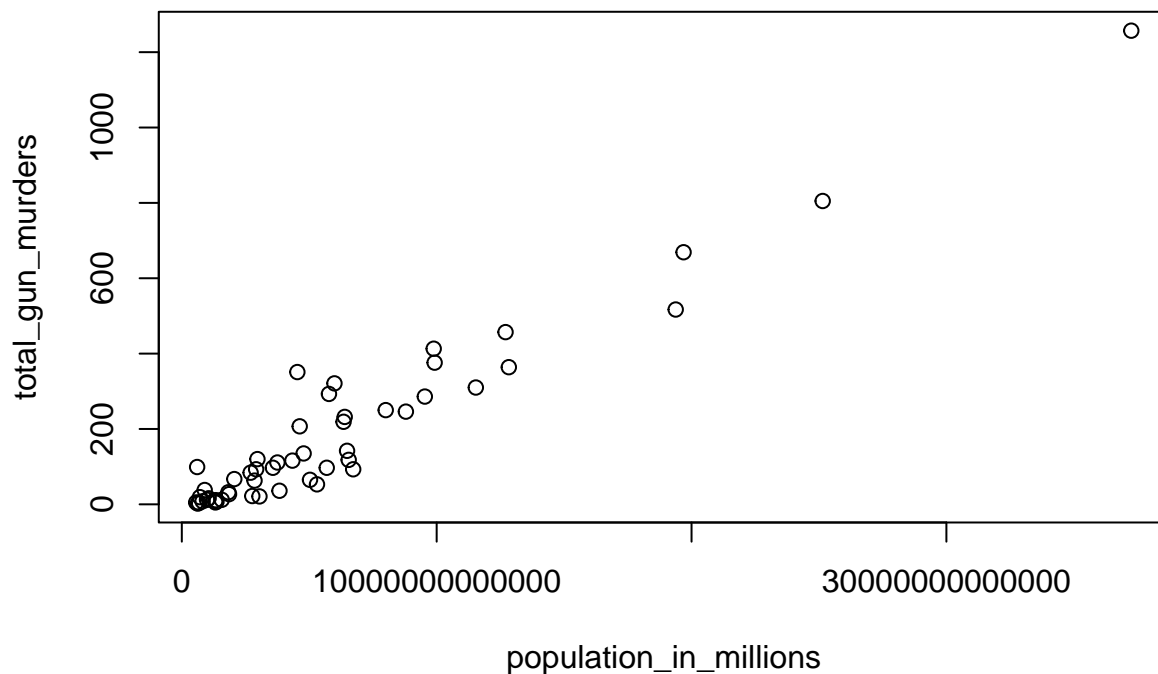
```
## [1] "MU"
```

43. Fizemos um gráfico do total de assassinatos versus população e notamos uma forte relação. Não é de surpreender que estados com populações maiores tenham tido mais assassinatos.

```
library(dslabs)

data("murders")

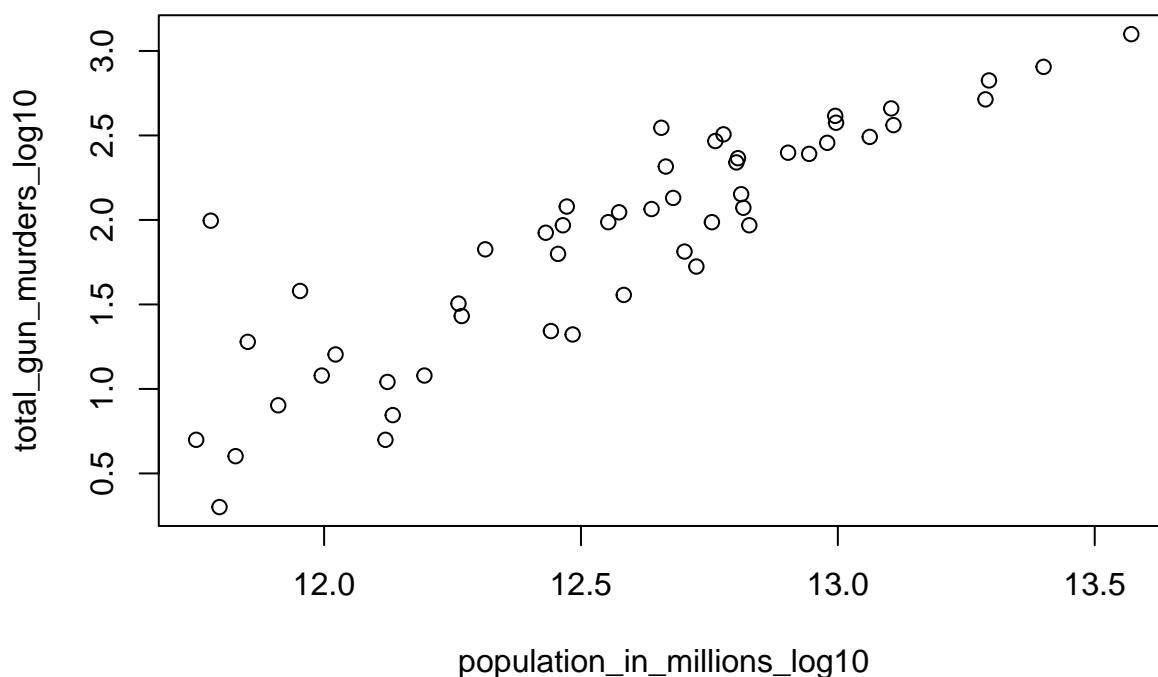
population_in_millions <- murders$population * 10^6
total_gun_murders <- murders$total
plot(population_in_millions, total_gun_murders)
```



Tenha em mente que muitos estados têm populações abaixo de 5 milhões e estão agrupados. Podemos obter mais insights fazendo este gráfico na escala logarítmica. Transforme as variáveis usando a transformação \log_{10} e depois represente-as.

```
library(dslabs)
data("murders")

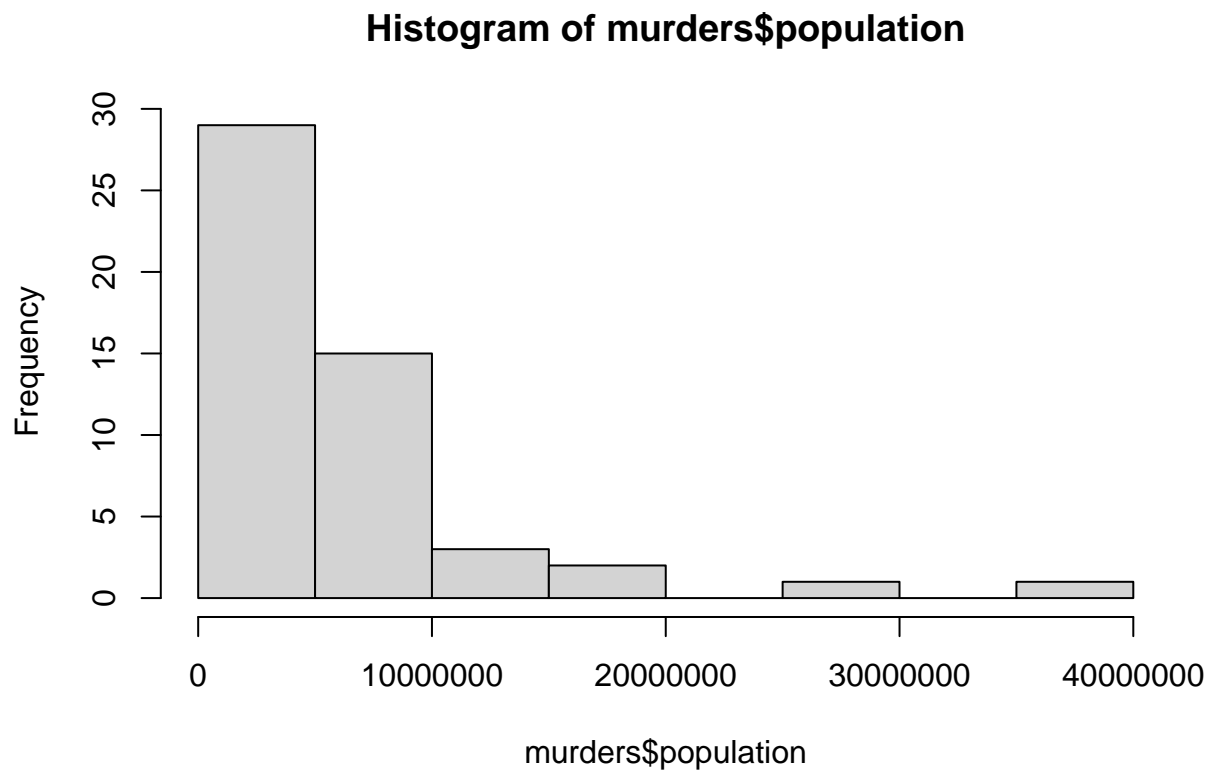
population_in_millions_log10 <- log10(murders$population * 10^6)
total_gun_murders_log10 <- log10(murders$total)
plot(population_in_millions_log10, total_gun_murders_log10)
```



44. Crie um histograma das populações do estado.

```
library(dslabs)
data("murders")

hist(murders$population)
```



45. Gere boxplots das populações estaduais por região.

```
library(dslabs)
data("murders")

boxplot(murders$population ~ murders$region)
```

