

# Análises Ecológicas no R: Exercícios e Soluções

2022-02-11



# Contents

Cap. 4 - Introdução ao R	7
Cap. 5 - Tidyverse	21
Cap. 15 - Dados geoespaciais no R	27



# Sobre

Aqui você encontra os **exercícios e soluções** do livro Análises Ecológicas no R.



## Cap. 4 - Introdução ao R

4.1 Use o R para verificar o resultado da operação  $7 + 7 \div 7 + 7 \times 7 - 7$ .

Solução:

```
7 + 7 / 7 + 7 * 7 - 7  
#> [1] 50
```

4.2 Verifique através do R se  $3 \times 2^3$  é maior que  $2 \times 3^2$ .

Solução:

```
3 * 2^3 > 2 * 3^2  
#> [1] TRUE
```

4.3 Crie dois objetos (qualquer nome) com os valores 100 e 300. Multiplique esses objetos (função `prod()`) e atribua ao objeto **mult**. Faça o logaritmo natural (função `log()`) do objeto **mult** e atribua ao objeto **ln**.

Solução:

```
obj1 <- 100  
obj2 <- 300  
mult <- prod(obj1, obj2)  
ln <- log(obj1, obj2)
```

4.4 Quantos pacotes existem no CRAN nesse momento? Execute essa combinação no Console: `nrow(available.packages(repos = "http://cran.r-project.org"))`.

Solução:

```
nrow(available.packages(repos = "http://cran.r-project.org"))  
#> [1] 18913
```

4.5 Instale o pacote **tidyverse** do CRAN.

Solução:

```
install.packages("tidyverse", dependencies = TRUE)
```

**4.6** Escolha números para jogar na mega-sena usando o R, nomeando o objeto como **mega**. Lembrando: são 6 valores de 1 a 60 e atribua a um objeto.

Solução:

```
mega <- sample(x = 1:60, size = 6, replace = FALSE)
mega
#> [1] 25 53 9 22 13 20
```

**4.7** Crie um fator chamado **tr**, com dois níveis (“cont” e “trat”) para descrever 100 locais de amostragem, 50 de cada tratamento. O fator deve ser dessa forma cont, cont, cont, ..., cont, trat, trat, ..., trat.

Solução:

```
tr <- factor(c(rep("cont", each = 50), rep("trat", each = 50)))
tr
#> [1] cont cont cont cont cont cont cont cont cont cont cont
#> [12] cont cont cont cont cont cont cont cont cont cont cont
#> [23] cont cont cont cont cont cont cont cont cont cont cont
#> [34] cont cont cont cont cont cont cont cont cont cont cont
#> [45] cont cont cont cont cont cont cont cont cont cont cont
#> [56] trat trat trat trat trat trat trat trat trat trat trat
#> [67] trat trat trat trat trat trat trat trat trat trat trat
#> [78] trat trat trat trat trat trat trat trat trat trat trat
#> [89] trat trat trat trat trat trat trat trat trat trat trat
#> [100] trat
#> Levels: cont trat
```

**4.8** Crie uma matriz chamada **ma**, resultante da disposição de um vetor composto por 1000 valores aleatórios entre 0 e 10. A matriz deve conter 100 linhas e ser disposta por colunas.

Solução:

```
ma <- matrix(sample(0:10, 1000, rep = TRUE), nrow = 100, byrow = FALSE)
ma
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#> [1,] 8 0 2 7 5 5 1 0 2 8
#> [2,] 0 10 4 9 7 7 7 3 5 0
#> [3,] 3 3 1 0 10 10 0 9 3 5
#> [4,] 5 0 0 2 8 10 0 9 4 2
#> [5,] 7 3 10 4 6 9 10 4 1 0
#> [6,] 2 10 6 9 6 6 3 3 4 6
#> [7,] 9 7 0 5 3 2 3 2 9 7
#> [8,] 5 7 8 7 10 7 0 10 0 4
#> [9,] 2 10 8 2 4 5 1 9 4 10
#> [10,] 10 7 0 2 3 5 5 6 5 7
#> [11,] 3 10 5 8 2 3 0 9 7 2
```



#> [12,]	9	9	1	5	9	7	7	2	3	4
#> [13,]	10	2	7	10	8	9	4	8	0	2
#> [14,]	0	9	4	9	9	7	10	6	2	6
#> [15,]	5	3	9	9	7	6	0	3	3	6
#> [16,]	7	0	5	10	7	4	7	6	9	9
#> [17,]	4	8	1	7	7	10	0	6	2	5
#> [18,]	6	10	5	2	3	5	10	10	5	10
#> [19,]	3	2	6	4	10	1	6	4	1	1
#> [20,]	3	10	5	8	0	4	9	4	6	2
#> [21,]	0	9	7	1	6	5	4	3	0	10
#> [22,]	2	1	6	0	7	2	6	7	7	7
#> [23,]	3	9	2	0	4	3	4	1	4	5
#> [24,]	5	7	6	9	4	5	2	2	5	2
#> [25,]	5	4	8	5	9	10	4	3	1	8
#> [26,]	7	2	3	6	4	3	2	5	10	1
#> [27,]	7	3	1	7	7	2	2	8	7	1
#> [28,]	10	7	10	1	8	4	1	6	9	3
#> [29,]	3	7	5	4	7	1	1	8	9	2
#> [30,]	0	2	5	6	6	10	9	6	5	0
#> [31,]	6	0	8	8	3	8	8	1	3	1
#> [32,]	4	9	1	9	9	6	7	8	8	3
#> [33,]	7	9	4	7	5	0	0	6	1	10
#> [34,]	6	4	1	0	0	3	8	2	1	0
#> [35,]	4	5	5	10	2	9	0	9	1	6
#> [36,]	7	8	5	10	8	3	4	2	9	7
#> [37,]	3	9	3	10	5	2	3	10	6	10
#> [38,]	8	9	0	10	2	5	2	2	8	2
#> [39,]	5	9	8	3	4	8	6	4	9	8
#> [40,]	3	2	8	0	7	6	5	5	1	0
#> [41,]	3	8	9	6	6	8	7	2	5	0
#> [42,]	5	1	6	7	8	7	7	6	8	1
#> [43,]	7	1	5	9	0	2	7	6	0	3
#> [44,]	1	3	9	10	2	9	10	3	6	8
#> [45,]	7	1	4	5	7	10	5	5	1	2
#> [46,]	0	7	0	10	5	2	4	1	3	9
#> [47,]	9	3	0	6	7	5	4	0	0	0
#> [48,]	8	8	4	2	9	9	4	8	1	7
#> [49,]	9	2	4	4	7	5	2	5	7	3
#> [50,]	3	0	8	1	3	3	3	1	4	0
#> [51,]	9	6	8	10	9	9	8	4	8	2
#> [52,]	4	9	10	9	7	1	8	7	5	10
#> [53,]	4	10	6	0	7	1	4	10	9	8
#> [54,]	7	4	0	6	7	0	0	0	0	7
#> [55,]	5	8	8	1	3	9	10	8	0	4
#> [56,]	3	9	1	4	4	9	10	8	5	9

#> [57,]	5	10	3	9	4	0	4	10	1	4
#> [58,]	2	0	3	1	3	6	3	0	3	6
#> [59,]	10	4	3	3	1	7	3	1	5	8
#> [60,]	9	4	4	9	2	4	4	10	3	9
#> [61,]	6	0	3	4	0	6	5	8	6	4
#> [62,]	8	1	2	0	0	0	5	7	0	9
#> [63,]	8	10	1	3	3	5	9	0	5	3
#> [64,]	10	9	0	5	5	6	7	4	5	2
#> [65,]	7	7	9	2	5	7	3	0	4	2
#> [66,]	5	2	7	6	7	3	4	2	6	7
#> [67,]	8	8	7	0	0	5	9	10	3	9
#> [68,]	9	2	0	9	3	3	5	10	7	0
#> [69,]	9	8	4	3	4	10	8	7	2	3
#> [70,]	3	0	7	4	6	1	7	7	7	4
#> [71,]	0	0	4	9	7	3	5	6	7	4
#> [72,]	9	7	8	4	9	8	10	10	10	0
#> [73,]	8	3	6	10	4	0	2	1	5	4
#> [74,]	0	10	6	7	9	0	4	6	8	2
#> [75,]	10	10	6	3	3	6	7	1	6	4
#> [76,]	9	6	9	0	1	7	3	0	9	3
#> [77,]	3	9	6	0	1	5	5	4	5	8
#> [78,]	6	6	9	6	0	7	6	7	10	4
#> [79,]	8	7	2	8	2	6	1	9	0	2
#> [80,]	6	5	2	1	8	4	1	10	1	5
#> [81,]	6	1	3	10	4	5	6	9	4	9
#> [82,]	3	4	3	6	0	0	5	4	0	3
#> [83,]	6	5	8	4	1	6	3	9	3	5
#> [84,]	8	5	1	0	9	9	1	7	10	8
#> [85,]	4	0	6	2	9	10	2	7	3	9
#> [86,]	2	0	1	7	9	2	2	6	6	3
#> [87,]	6	2	1	9	8	6	2	0	4	5
#> [88,]	7	2	1	10	9	7	1	3	4	1
#> [89,]	5	1	0	1	5	1	10	9	6	7
#> [90,]	10	0	5	2	7	6	1	0	5	1
#> [91,]	6	4	7	2	10	4	8	10	6	5
#> [92,]	0	10	8	5	4	3	8	10	5	9
#> [93,]	5	3	9	6	9	0	6	1	0	9
#> [94,]	10	0	5	9	3	10	6	4	7	1
#> [95,]	6	6	1	1	7	4	3	1	4	3
#> [96,]	7	4	1	9	8	10	1	1	10	1
#> [97,]	9	1	0	4	6	10	2	2	6	2
#> [98,]	2	4	0	5	6	4	5	6	5	2
#> [99,]	5	1	4	10	6	8	3	5	4	0
#> [100,]	7	1	3	2	10	10	9	8	2	3

4.9 Crie um data frame chamado **df**, resultante da composição dos vetores:

1. id: 1:50
2. sp: sp01, sp02, ..., sp49, sp50
3. ab: 50 valores aleatórios entre 0 a 5

Solução:

```
df <- data.frame(id = 1:50,
                 sp = c(paste0("sp0", 1:9), paste0("sp", 10:50)),
                 ab = sample(0:5, 50, rep = TRUE))

df
#>      id    sp ab
#>  1     1 sp01  0
#>  2     2 sp02  3
#>  3     3 sp03  3
#>  4     4 sp04  5
#>  5     5 sp05  4
#>  6     6 sp06  1
#>  7     7 sp07  2
#>  8     8 sp08  0
#>  9     9 sp09  0
#> 10    10 sp10  3
#> 11    11 sp11  1
#> 12    12 sp12  2
#> 13    13 sp13  3
#> 14    14 sp14  0
#> 15    15 sp15  2
#> 16    16 sp16  5
#> 17    17 sp17  4
#> 18    18 sp18  2
#> 19    19 sp19  5
#> 20    20 sp20  3
#> 21    21 sp21  3
#> 22    22 sp22  5
#> 23    23 sp23  4
#> 24    24 sp24  4
#> 25    25 sp25  3
#> 26    26 sp26  4
#> 27    27 sp27  3
#> 28    28 sp28  4
#> 29    29 sp29  0
#> 30    30 sp30  0
#> 31    31 sp31  0
#> 32    32 sp32  5
#> 33    33 sp33  5
#> 34    34 sp34  2
```

```
#> 35 35 sp35 3
#> 36 36 sp36 5
#> 37 37 sp37 2
#> 38 38 sp38 4
#> 39 39 sp39 4
#> 40 40 sp40 2
#> 41 41 sp41 4
#> 42 42 sp42 3
#> 43 43 sp43 3
#> 44 44 sp44 4
#> 45 45 sp45 5
#> 46 46 sp46 0
#> 47 47 sp47 2
#> 48 48 sp48 1
#> 49 49 sp49 1
#> 50 50 sp50 0
```

**4.10** Crie uma lista com os objetos criados anteriormente: **mega**, **tr**, **ma** e **df**.

Solução:

```
lis <- list(mega, tr, ma, df)
lis
#> [[1]]
#> [1] 25 53 9 22 13 20
#>
#> [[2]]
#> [1] cont cont cont cont cont cont cont cont cont cont cont
#> [12] cont cont cont cont cont cont cont cont cont cont cont
#> [23] cont cont cont cont cont cont cont cont cont cont cont
#> [34] cont cont cont cont cont cont cont cont cont cont cont
#> [45] cont cont cont cont cont cont cont trat trat trat trat
#> [56] trat trat trat trat trat trat trat trat trat trat trat
#> [67] trat trat trat trat trat trat trat trat trat trat trat
#> [78] trat trat trat trat trat trat trat trat trat trat trat
#> [89] trat trat trat trat trat trat trat trat trat trat trat
#> [100] trat
#> Levels: cont trat
#>
#> [[3]]
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#> [1,]    8    0    2    7    5    5    1    0    2    8
#> [2,]    0   10    4    9    7    7    7    3    5    0
#> [3,]    3    3    1    0   10   10    0    9    3    5
#> [4,]    5    0    0    2    8   10    0    9    4    2
#> [5,]    7    3   10    4    6    9   10    4    1    0
```

#>	[6,]	2	10	6	9	6	6	3	3	4	6
#>	[7,]	9	7	0	5	3	2	3	2	9	7
#>	[8,]	5	7	8	7	10	7	0	10	0	4
#>	[9,]	2	10	8	2	4	5	1	9	4	10
#>	[10,]	10	7	0	2	3	5	5	6	5	7
#>	[11,]	3	10	5	8	2	3	0	9	7	2
#>	[12,]	9	9	1	5	9	7	7	2	3	4
#>	[13,]	10	2	7	10	8	9	4	8	0	2
#>	[14,]	0	9	4	9	9	7	10	6	2	6
#>	[15,]	5	3	9	9	7	6	0	3	3	6
#>	[16,]	7	0	5	10	7	4	7	6	9	9
#>	[17,]	4	8	1	7	7	10	0	6	2	5
#>	[18,]	6	10	5	2	3	5	10	10	5	10
#>	[19,]	3	2	6	4	10	1	6	4	1	1
#>	[20,]	3	10	5	8	0	4	9	4	6	2
#>	[21,]	0	9	7	1	6	5	4	3	0	10
#>	[22,]	2	1	6	0	7	2	6	7	7	7
#>	[23,]	3	9	2	0	4	3	4	1	4	5
#>	[24,]	5	7	6	9	4	5	2	2	5	2
#>	[25,]	5	4	8	5	9	10	4	3	1	8
#>	[26,]	7	2	3	6	4	3	2	5	10	1
#>	[27,]	7	3	1	7	7	2	2	8	7	1
#>	[28,]	10	7	10	1	8	4	1	6	9	3
#>	[29,]	3	7	5	4	7	1	1	8	9	2
#>	[30,]	0	2	5	6	6	10	9	6	5	0
#>	[31,]	6	0	8	8	3	8	8	1	3	1
#>	[32,]	4	9	1	9	9	6	7	8	8	3
#>	[33,]	7	9	4	7	5	0	0	6	1	10
#>	[34,]	6	4	1	0	0	3	8	2	1	0
#>	[35,]	4	5	5	10	2	9	0	9	1	6
#>	[36,]	7	8	5	10	8	3	4	2	9	7
#>	[37,]	3	9	3	10	5	2	3	10	6	10
#>	[38,]	8	9	0	10	2	5	2	2	8	2
#>	[39,]	5	9	8	3	4	8	6	4	9	8
#>	[40,]	3	2	8	0	7	6	5	5	1	0
#>	[41,]	3	8	9	6	6	8	7	2	5	0
#>	[42,]	5	1	6	7	8	7	7	6	8	1
#>	[43,]	7	1	5	9	0	2	7	6	0	3
#>	[44,]	1	3	9	10	2	9	10	3	6	8
#>	[45,]	7	1	4	5	7	10	5	5	1	2
#>	[46,]	0	7	0	10	5	2	4	1	3	9
#>	[47,]	9	3	0	6	7	5	4	0	0	0
#>	[48,]	8	8	4	2	9	9	4	8	1	7
#>	[49,]	9	2	4	4	7	5	2	5	7	3
#>	[50,]	3	0	8	1	3	3	3	1	4	0

#> [51,]	9	6	8	10	9	9	8	4	8	2
#> [52,]	4	9	10	9	7	1	8	7	5	10
#> [53,]	4	10	6	0	7	1	4	10	9	8
#> [54,]	7	4	0	6	7	0	0	0	0	7
#> [55,]	5	8	8	1	3	9	10	8	0	4
#> [56,]	3	9	1	4	4	9	10	8	5	9
#> [57,]	5	10	3	9	4	0	4	10	1	4
#> [58,]	2	0	3	1	3	6	3	0	3	6
#> [59,]	10	4	3	3	1	7	3	1	5	8
#> [60,]	9	4	4	9	2	4	4	10	3	9
#> [61,]	6	0	3	4	0	6	5	8	6	4
#> [62,]	8	1	2	0	0	0	5	7	0	9
#> [63,]	8	10	1	3	3	5	9	0	5	3
#> [64,]	10	9	0	5	5	6	7	4	5	2
#> [65,]	7	7	9	2	5	7	3	0	4	2
#> [66,]	5	2	7	6	7	3	4	2	6	7
#> [67,]	8	8	7	0	0	5	9	10	3	9
#> [68,]	9	2	0	9	3	3	5	10	7	0
#> [69,]	9	8	4	3	4	10	8	7	2	3
#> [70,]	3	0	7	4	6	1	7	7	7	4
#> [71,]	0	0	4	9	7	3	5	6	7	4
#> [72,]	9	7	8	4	9	8	10	10	10	0
#> [73,]	8	3	6	10	4	0	2	1	5	4
#> [74,]	0	10	6	7	9	0	4	6	8	2
#> [75,]	10	10	6	3	3	6	7	1	6	4
#> [76,]	9	6	9	0	1	7	3	0	9	3
#> [77,]	3	9	6	0	1	5	5	4	5	8
#> [78,]	6	6	9	6	0	7	6	7	10	4
#> [79,]	8	7	2	8	2	6	1	9	0	2
#> [80,]	6	5	2	1	8	4	1	10	1	5
#> [81,]	6	1	3	10	4	5	6	9	4	9
#> [82,]	3	4	3	6	0	0	5	4	0	3
#> [83,]	6	5	8	4	1	6	3	9	3	5
#> [84,]	8	5	1	0	9	9	1	7	10	8
#> [85,]	4	0	6	2	9	10	2	7	3	9
#> [86,]	2	0	1	7	9	2	2	6	6	3
#> [87,]	6	2	1	9	8	6	2	0	4	5
#> [88,]	7	2	1	10	9	7	1	3	4	1
#> [89,]	5	1	0	1	5	1	10	9	6	7
#> [90,]	10	0	5	2	7	6	1	0	5	1
#> [91,]	6	4	7	2	10	4	8	10	6	5
#> [92,]	0	10	8	5	4	3	8	10	5	9
#> [93,]	5	3	9	6	9	0	6	1	0	9
#> [94,]	10	0	5	9	3	10	6	4	7	1
#> [95,]	6	6	1	1	7	4	3	1	4	3

```

#> [96,] 7 4 1 9 8 10 1 1 10 1
#> [97,] 9 1 0 4 6 10 2 2 6 2
#> [98,] 2 4 0 5 6 4 5 6 5 2
#> [99,] 5 1 4 10 6 8 3 5 4 0
#> [100,] 7 1 3 2 10 10 9 8 2 3
#>
#> [[4]]
#> id sp ab
#> 1 1 sp01 0
#> 2 2 sp02 3
#> 3 3 sp03 3
#> 4 4 sp04 5
#> 5 5 sp05 4
#> 6 6 sp06 1
#> 7 7 sp07 2
#> 8 8 sp08 0
#> 9 9 sp09 0
#> 10 10 sp10 3
#> 11 11 sp11 1
#> 12 12 sp12 2
#> 13 13 sp13 3
#> 14 14 sp14 0
#> 15 15 sp15 2
#> 16 16 sp16 5
#> 17 17 sp17 4
#> 18 18 sp18 2
#> 19 19 sp19 5
#> 20 20 sp20 3
#> 21 21 sp21 3
#> 22 22 sp22 5
#> 23 23 sp23 4
#> 24 24 sp24 4
#> 25 25 sp25 3
#> 26 26 sp26 4
#> 27 27 sp27 3
#> 28 28 sp28 4
#> 29 29 sp29 0
#> 30 30 sp30 0
#> 31 31 sp31 0
#> 32 32 sp32 5
#> 33 33 sp33 5
#> 34 34 sp34 2
#> 35 35 sp35 3
#> 36 36 sp36 5
#> 37 37 sp37 2

```

```
#> 38 38 sp38 4
#> 39 39 sp39 4
#> 40 40 sp40 2
#> 41 41 sp41 4
#> 42 42 sp42 3
#> 43 43 sp43 3
#> 44 44 sp44 4
#> 45 45 sp45 5
#> 46 46 sp46 0
#> 47 47 sp47 2
#> 48 48 sp48 1
#> 49 49 sp49 1
#> 50 50 sp50 0
```

**4.11** Selecione os elementos ímpares do objeto **tr** e atribua ao objeto **tr\_impar**.

Solução:

```
tr_impar <- tr[seq(1, 99, 2)]
tr_impar
#> [1] cont cont cont cont cont cont cont cont cont cont cont
#> [12] cont cont cont cont cont cont cont cont cont cont cont
#> [23] cont cont cont trat trat trat trat trat trat trat trat
#> [34] trat trat trat trat trat trat trat trat trat trat trat
#> [45] trat trat trat trat trat trat
#> Levels: cont trat
```

**4.12** Selecione as linhas com ids pares do objeto **df** e atribua ao objeto **df\_ids\_par**.

Solução:

```
df_ids_par <- df[seq(2, 100, 2), ]
df_ids_par
#>      id  sp ab
#> 2      2 sp02 3
#> 4      4 sp04 5
#> 6      6 sp06 1
#> 8      8 sp08 0
#> 10     10 sp10 3
#> 12     12 sp12 2
#> 14     14 sp14 0
#> 16     16 sp16 5
#> 18     18 sp18 2
#> 20     20 sp20 3
#> 22     22 sp22 5
#> 24     24 sp24 4
#> 26     26 sp26 4
```



```
#> 28      28 sp28  4
#> 30      30 sp30  0
#> 32      32 sp32  5
#> 34      34 sp34  2
#> 36      36 sp36  5
#> 38      38 sp38  4
#> 40      40 sp40  2
#> 42      42 sp42  3
#> 44      44 sp44  4
#> 46      46 sp46  0
#> 48      48 sp48  1
#> 50      50 sp50  0
#> NA      NA <NA> NA
#> NA.1    NA <NA> NA
#> NA.2    NA <NA> NA
#> NA.3    NA <NA> NA
#> NA.4    NA <NA> NA
#> NA.5    NA <NA> NA
#> NA.6    NA <NA> NA
#> NA.7    NA <NA> NA
#> NA.8    NA <NA> NA
#> NA.9    NA <NA> NA
#> NA.10   NA <NA> NA
#> NA.11   NA <NA> NA
#> NA.12   NA <NA> NA
#> NA.13   NA <NA> NA
#> NA.14   NA <NA> NA
#> NA.15   NA <NA> NA
#> NA.16   NA <NA> NA
#> NA.17   NA <NA> NA
#> NA.18   NA <NA> NA
#> NA.19   NA <NA> NA
#> NA.20   NA <NA> NA
#> NA.21   NA <NA> NA
#> NA.22   NA <NA> NA
#> NA.23   NA <NA> NA
#> NA.24   NA <NA> NA
```

**4.13** Faça uma amostragem de 10 linhas do objeto **df** e atribua ao objeto **df\_amos10**.

Solução:

```
df_amos10 <- df[sample(nrow(df), 10), ]
df_amos10
#>      id  sp ab
```

```
#> 37 37 sp37 2
#> 36 36 sp36 5
#> 25 25 sp25 3
#> 42 42 sp42 3
#> 30 30 sp30 0
#> 6 6 sp06 1
#> 44 44 sp44 4
#> 33 33 sp33 5
#> 32 32 sp32 5
#> 17 17 sp17 4
```

**4.14** Amostre 10 linhas do objeto **ma**, mas utilizando as linhas amostradas do **df\_amos10** e atribua ao objeto **ma\_amos10**.

Solução:

```
ma_amos10 <- ma[df_amos10$id, ]
ma_amos10
#>      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#> [1,]    3    9    3   10    5    2    3   10    6   10
#> [2,]    7    8    5   10    8    3    4    2    9    7
#> [3,]    5    4    8    5    9   10    4    3    1    8
#> [4,]    5    1    6    7    8    7    7    6    8    1
#> [5,]    0    2    5    6    6   10    9    6    5    0
#> [6,]    2   10    6    9    6    6    3    3    4    6
#> [7,]    1    3    9   10    2    9   10    3    6    8
#> [8,]    7    9    4    7    5    0    0    6    1   10
#> [9,]    4    9    1    9    9    6    7    8    8    3
#> [10,]   4    8    1    7    7   10    0    6    2    5
```

**4.15** Una as colunas dos objetos **df\_amos10** e **ma\_amos10** e atribua ao objeto **dados\_amos10**.

Solução:

```
dados_amos10 <- cbind(df_amos10, ma_amos10)
dados_amos10
#>      id  sp ab 1  2 3  4 5  6  7  8 9 10
#> 37 37 sp37 2 3  9 3 10 5  2  3 10 6 10
#> 36 36 sp36 5 7  8 5 10 8  3  4  2 9  7
#> 25 25 sp25 3 5  4 8  5 9 10  4  3 1  8
#> 42 42 sp42 3 5  1 6  7 8  7  7  6 8  1
#> 30 30 sp30 0 0  2 5  6 6 10  9  6 5  0
#> 6 6 sp06 1 2 10 6  9 6  6  3  3 4  6
#> 44 44 sp44 4 1  3 9 10 2  9 10  3 6  8
#> 33 33 sp33 5 7  9 4  7 5  0  0  6 1 10
#> 32 32 sp32 5 4  9 1  9 9  6  7  8 8  3
```

#> 17 17 sp17 4 4 8 1 7 7 10 0 6 2 5



## Cap. 5 - Tidyverse

**5.1** Reescreva as operações abaixo utilizando pipes %>%. - log10(cumsum(1:100))  
- sum(sqrt(abs(rnorm(100)))) - sum(sort(sample(1:10, 10000, rep = TRUE)))

Solução:

```
library(tidyverse)
#> -- Attaching packages ----- tidyverse 1.3.1 --
#> v ggplot2 3.3.5      v purrr 0.3.4
#> v tibble 3.1.6       v dplyr 1.0.7
#> v tidyr 1.1.4        v stringr 1.4.0
#> v readr 2.1.1       v forcats 0.5.1
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()     masks stats::lag()

1:100 %>%
  cumsum() %>%
  log10()
#> [1] 0.0000000 0.4771213 0.7781513 1.0000000 1.1760913
#> [6] 1.3222193 1.4471580 1.5563025 1.6532125 1.7403627
#> [11] 1.8195439 1.8920946 1.9590414 2.0211893 2.0791812
#> [16] 2.1335389 2.1846914 2.2329961 2.2787536 2.3222193
#> [21] 2.3636120 2.4031205 2.4409091 2.4771213 2.5118834
#> [26] 2.5453071 2.5774918 2.6085260 2.6384893 2.6674530
#> [31] 2.6954817 2.7226339 2.7489629 2.7745170 2.7993405
#> [36] 2.8234742 2.8469553 2.8698182 2.8920946 2.9138139
#> [41] 2.9350032 2.9556878 2.9758911 2.9956352 3.0149403
#> [46] 3.0338257 3.0523091 3.0704073 3.0881361 3.1055102
#> [51] 3.1225435 3.1392492 3.1556396 3.1717265 3.1875207
#> [56] 3.2030329 3.2182729 3.2332500 3.2479733 3.2624511
#> [61] 3.2766915 3.2907022 3.3044905 3.3180633 3.3314273
#> [66] 3.3445887 3.3575537 3.3703280 3.3829171 3.3953264
#> [71] 3.4075608 3.4196254 3.4315246 3.4432630 3.4548449
```

```
#> [76] 3.4662743 3.4775553 3.4886917 3.4996871 3.5105450
#> [81] 3.5212689 3.5318619 3.5423274 3.5526682 3.5628874
#> [86] 3.5729877 3.5829719 3.5928427 3.6026025 3.6122539
#> [91] 3.6217992 3.6312408 3.6405808 3.6498215 3.6589648
#> [96] 3.6680130 3.6769678 3.6858313 3.6946052 3.7032914

rnorm(100) %>%
  abs() %>%
  sqrt() %>%
  sum()
#> [1] 82.79655

sample(1:10, 10000, rep = TRUE) %>%
  sort() %>%
  sum()
#> [1] 54789
```

**5.2** Use a função `download.file()` e `unzip()` para baixar e extrair o arquivo do data paper de médios e grandes mamíferos: ATLANTIC MAMMALS. Em seguida, importe para o R, usando a função `readr::read_csv()`.

Solução:

```
library(tidyverse)
download.file(url = "https://esajournals.onlinelibrary.wiley.com/action/downloadSupplement?doi=10.1111%2Fesaj.12785", destfile = "ecy2785-sup-0001-DataS1.zip", mode = "wb")

unzip("ecy2785-sup-0001-DataS1.zip")

dp_lm <- readr::read_csv("ATLANTIC_MAMMAL_MID_LARGE_assemblages_and_sites.csv")
#> Warning: One or more parsing issues, see `problems()` for details
#> Rows: 4680 Columns: 40
#> -- Column specification -----
#> Delimiter: ","
#> chr (27): ID, Country, State, Municipality, Study_locati...
#> dbl (11): Reference_paper_number, Publication_year, Year...
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

**5.3** Use a função `tibble::glimpse()` para ter uma noção geral dos dados importados no item anterior.

Solução:

```

library(tidyverse)
dplyr::glimpse(dp_lm)
#> Rows: 4,680
#> Columns: 40
#> $ ID <chr> "AML01", "AML01", "AML01", ~
#> $ Reference_paper_number <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
#> $ Country <chr> "Brazil", "Brazil", "Brazil~
#> $ State <chr> "rio_grande_do_sul", "rio_g~
#> $ Municipality <chr> "Sinimbu", "Sinimbu", "Sini~
#> $ Study_location <chr> "Reserva Particular do Patr~
#> $ Latitude <dbl> -29.38333, -29.38333, -29.3~
#> $ Longitude <dbl> -52.53333, -52.53333, -52.5~
#> $ Precision <chr> "not_precise", "not_precise~
#> $ Size_ha <chr> "221", "221", "221", "221",~
#> $ Temperature <chr> "18", "18", "18", "18", "18~
#> $ Altitude <chr> "150-650", "150-650", "150~~
#> $ Annual_rainfall <chr> NA, NA, NA, NA, NA, NA, NA,~
#> $ Vegetation_type <chr> "Semideciduous forest", "Se~
#> $ Protect_area <chr> "yes", "yes", "yes", "yes",~
#> $ Matrix <chr> NA, NA, NA, NA, NA, NA, NA,~
#> $ Reference <chr> "Abreu-Junior, E.F. and Koh~
#> $ Publication_year <dbl> 2009, 2009, 2009, 2009, 200~
#> $ Type_of_publication <chr> "Article", "Article", "Arti~
#> $ Month_start <chr> "November", "November", "No~
#> $ Year_start <dbl> 2007, 2007, 2007, 2007, 200~
#> $ Month_finish <chr> "April", "April", "April", ~
#> $ Year_finish <dbl> 2009, 2009, 2009, 2009, 200~
#> $ Total_of_months <dbl> 6, 6, 6, 6, 6, 6, 6, 6, ~
#> $ Sampling_habitat <chr> "Interior", "Interior", "In~
#> $ Effort <dbl> 109.00, 109.00, 109.00, 109~
#> $ Effort_method <chr> "camera_days", "camera_days~
#> $ Method <chr> "mixed_method", "mixed_meth~
#> $ Order <chr> "Carnivora", "Rodentia", "C~
#> $ Genus_on_paper <chr> "Cerdocyon", "Cuniculus", "~
#> $ Species_name_on_paper <chr> "Cerdocyon thous", "Cunicul~
#> $ Actual_species_Name <chr> "Cerdocyon thous", "Cunicul~
#> $ Number_of_record <chr> NA, NA, NA, NA, NA, NA, NA,~
#> $ `Density(groups/km2)` <dbl> NA, NA, NA, NA, NA, NA, NA,~
#> $ `Density(ind/km2)` <chr> NA, NA, NA, NA, NA, NA, NA,~
#> $ `Density(ind/km10)` <dbl> NA, NA, NA, NA, NA, NA, NA,~
#> $ `Abundance(%)` <dbl> NA, NA, NA, NA, NA, NA, NA,~
#> $ Abundance_relative <dbl> NA, NA, NA, NA, NA, NA, NA,~
#> $ `Abundance(10/km)` <dbl> NA, NA, NA, NA, NA, NA, NA,~
#> $ Voucher_Specimens <chr> NA, NA, NA, NA, NA, NA, NA,~

```

**5.4** Compare os dados de penguins (*palmerpenguins::penguins\_raw* e *palmerpenguins::penguins*). Monte uma série de funções dos pacotes *tidyr* e *dplyr* para limpar os dados e fazer com que o primeiro dado seja igual ao segundo.

Solução:

```
library(tidyverse)
library(palmerpenguins)

penguins_raw
#> # A tibble: 344 x 17
#>   studyName `Sample Number` Species      Region Island Stage
#>   <chr>          <dbl> <chr>      <chr> <chr> <chr>
#> 1 PAL0708              1 Adelie Pe~ Anvers Torge~ Adult~
#> 2 PAL0708              2 Adelie Pe~ Anvers Torge~ Adult~
#> 3 PAL0708              3 Adelie Pe~ Anvers Torge~ Adult~
#> 4 PAL0708              4 Adelie Pe~ Anvers Torge~ Adult~
#> 5 PAL0708              5 Adelie Pe~ Anvers Torge~ Adult~
#> 6 PAL0708              6 Adelie Pe~ Anvers Torge~ Adult~
#> 7 PAL0708              7 Adelie Pe~ Anvers Torge~ Adult~
#> 8 PAL0708              8 Adelie Pe~ Anvers Torge~ Adult~
#> 9 PAL0708              9 Adelie Pe~ Anvers Torge~ Adult~
#> 10 PAL0708             10 Adelie Pe~ Anvers Torge~ Adult~
#> # ... with 334 more rows, and 11 more variables:
#> #   Individual ID <chr>, Clutch Completion <chr>,
#> #   Date Egg <date>, Culmen Length (mm) <dbl>,
#> #   Culmen Depth (mm) <dbl>, Flipper Length (mm) <dbl>,
#> #   Body Mass (g) <dbl>, Sex <chr>,
#> #   Delta 15 N (o/oo) <dbl>, Delta 13 C (o/oo) <dbl>,
#> #   Comments <chr>

penguins
#> # A tibble: 344 x 8
#>   species island  bill_length_mm bill_depth_mm
#>   <fct>   <fct>         <dbl>         <dbl>
#> 1 Adelie  Torgersen     39.1           18.7
#> 2 Adelie  Torgersen     39.5           17.4
#> 3 Adelie  Torgersen     40.3            18
#> 4 Adelie  Torgersen      NA             NA
#> 5 Adelie  Torgersen     36.7           19.3
#> 6 Adelie  Torgersen     39.3           20.6
#> 7 Adelie  Torgersen     38.9           17.8
#> 8 Adelie  Torgersen     39.2           19.6
#> 9 Adelie  Torgersen     34.1           18.1
#> 10 Adelie Torgersen      42            20.2
#> # ... with 334 more rows, and 4 more variables:
#> #   flipper_length_mm <int>, body_mass_g <int>, sex <fct>,
```



```
#> #   year <int>

penguins_raw %>%
  dplyr::select(Species, Island, `Culmen Length (mm)`:Sex, `Date Egg`) %>%
  dplyr::rename(species = Species,
                island = Island,
                bill_length_mm = `Culmen Length (mm)`,
                bill_depth_mm = `Culmen Depth (mm)`,
                flipper_length_mm = `Flipper Length (mm)`,
                body_mass_g = `Body Mass (g)`,
                sex = Sex,
                year = `Date Egg`) %>%
  tidyr::separate(species, c("species", NA, NA, NA, NA)) %>%
  dplyr::mutate(sex = stringr::str_to_lower(sex),
                year = lubridate::year(year))

#> # A tibble: 344 x 8
#>   species island   bill_length_mm bill_depth_mm
#>   <chr>   <chr>         <dbl>         <dbl>
#> 1 Adelie  Torgersen         39.1          18.7
#> 2 Adelie  Torgersen         39.5          17.4
#> 3 Adelie  Torgersen         40.3           18
#> 4 Adelie  Torgersen          NA           NA
#> 5 Adelie  Torgersen         36.7          19.3
#> 6 Adelie  Torgersen         39.3          20.6
#> 7 Adelie  Torgersen         38.9          17.8
#> 8 Adelie  Torgersen         39.2          19.6
#> 9 Adelie  Torgersen         34.1          18.1
#> 10 Adelie Torgersen         42           20.2
#> # ... with 334 more rows, and 4 more variables:
#> #   flipper_length_mm <dbl>, body_mass_g <dbl>, sex <chr>,
#> #   year <dbl>
```

**5.5** Usando os dados de penguins (*palmerpenguins::penguins*), calcule a correlação de Pearson entre comprimento e profundidade do bico para cada espécie e para todas as espécies. Compare os índices de correlação para exemplificar o Paradoxo de Simpson.

Solução:

```
library(tidyverse)
library(palmerpenguins)

cor(penguins$bill_length_mm, penguins$bill_depth_mm, use = "na.or.complete")
#> [1] -0.2350529

penguins %>%
```

```

  dplyr::group_split(species) %>%
  purrr::map(~cor(.x$bill_length_mm, .x$bill_depth_mm, use = "na.or.complete"))
#> [[1]]
#> [1] 0.3914917
#>
#> [[2]]
#> [1] 0.6535362
#>
#> [[3]]
#> [1] 0.6433839

```

**5.6** Oficialmente a pandemia de COVID-19 começou no Brasil com o primeiro caso no dia 26 de fevereiro de 2020. Calcule quantos anos, meses e dias se passou desde então. Calcule também quanto tempo se passou até você ser vacinado.

Solução:

```

covid_inicio_br <- lubridate::dmy("26-02-2020")
vacina <- lubridate::dmy("20-07-2021")

intervalo_covid <- lubridate::interval(covid_inicio_br, lubridate::today())
intervalo_vacina <- lubridate::interval(covid_inicio_br, vacina)

lubridate::as.period(intervalo_covid)
#> [1] "1y 11m 16d 0H 0M 0S"
lubridate::as.period(intervalo_vacina)
#> [1] "1y 4m 24d 0H 0M 0S"

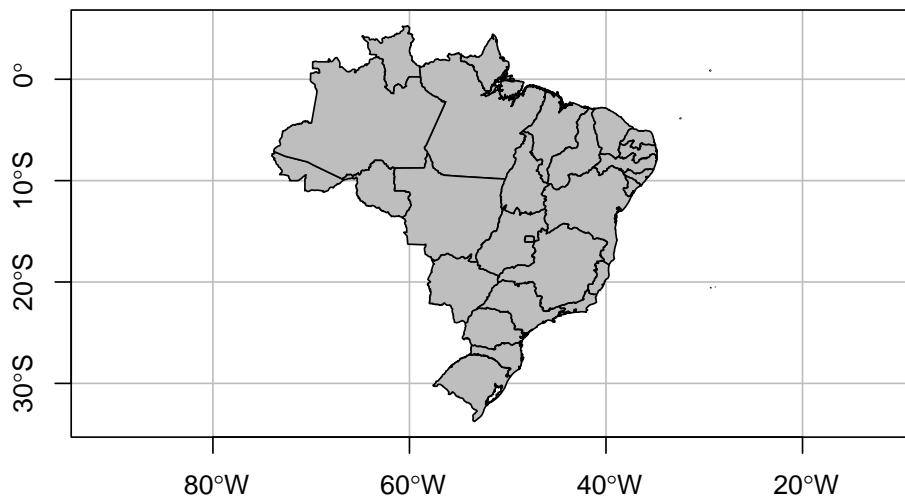
```

## Cap. 15 - Dados geoespaciais no R

**15.1** Importe o limite dos estados brasileiros no formato **sf** com o nome **br**. Para isso, use a função **ne\_states** do pacote **rnaturalearth**. Crie um mapa simples cinza utilizando a função **plot()**, selecionando a coluna **geometry** com o operador **\$** e com os argumentos **axes** e **graticule** verdadeiros.

Solução:

```
library(rnaturalearth)
br <- rnaturalearth::ne_states(country = "Brazil", returnclass = "sf")
plot(br$geometry, col = "gray", axes = TRUE, graticule = TRUE)
```



**15.2** Dados vetoriais podem ser criados com diversos erros de topologia, e.g., sobreposição de linhas ou polígonos ou buracos. Algumas funções exigem que os objetos vetoriais aos quais são atribuídos esses dados não possuam esses erros para que o algoritmo funcione. Para verificar se há erros, podemos usar a função **st\_is\_valid()** do pacote **sf**. Há diversas forma de correções desses erros,

mas vamos usar uma correção simples do R, com a função `st_make_valid()`. Vamos fazer essa correção para o `br` importado anteriormente e atribuindo ao objeto `br_valid`. Podemos conferir para saber se há erros e fazer um plot.

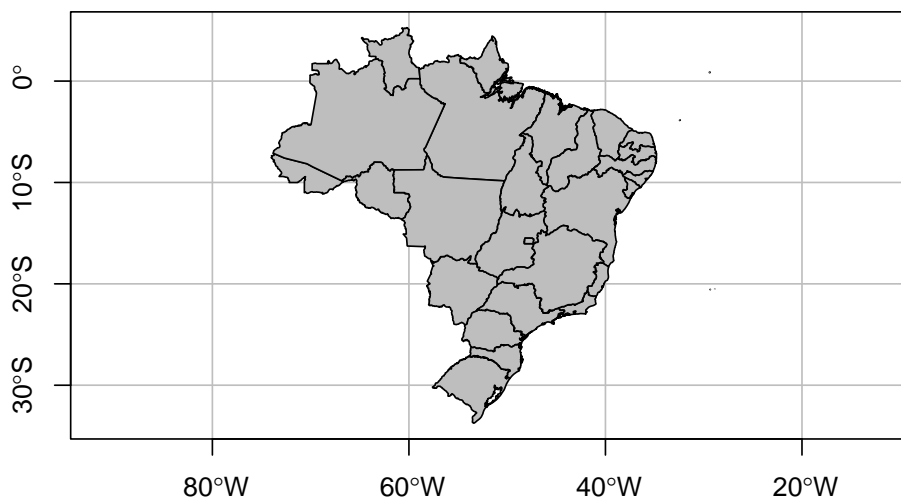
Solução:

```
library(sf)

sf::st_is_valid(br)
#> [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
#> [10] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
#> [19] TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE

br_valid <- sf::st_make_valid(br)
sf::st_is_valid(br_valid)
#> [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
#> [12] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
#> [23] TRUE TRUE TRUE TRUE TRUE

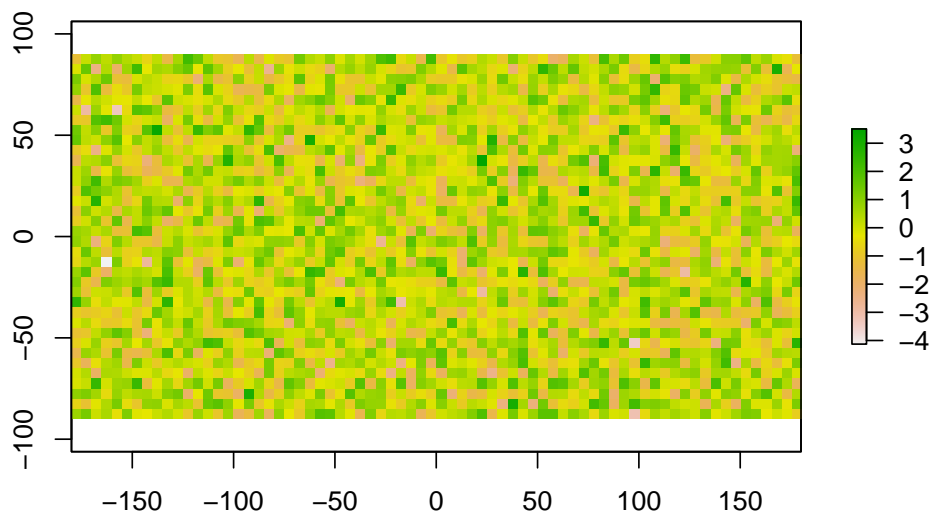
plot(br_valid$geometry, col = "gray", axes = TRUE, graticule = TRUE)
```



**15.3** Crie um objeto `RasterLayer` vazio chamado `ra` com resolução: de 5° (~600 km). Atribua um sistema de referência de coordenadas com o código **4326**. Atribua valores aleatórios de uma distribuição normal e plote o mesmo.

Solução:

```
library(raster)
ra <- raster::raster(res = 5, crs = 4326)
raster::values(ra) <- rnorm(raster::ncell(ra))
plot(ra)
```

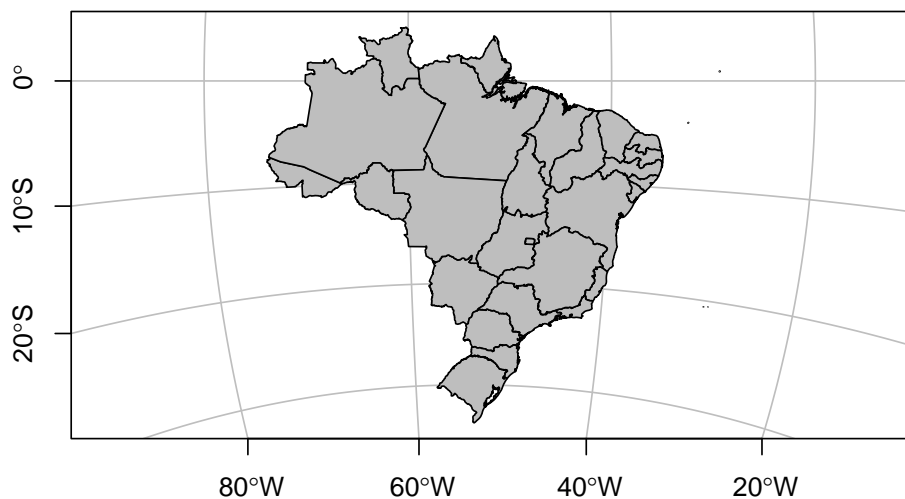


**15.4** Reprojeite o limite dos estados brasileiros do exercício anterior para o CRS SIRGAS 2000/Brazil Polyconic, utilizando o código EPSG:5880 e chamando de **br\_poly**. Faça um mapa simples como no exercício 1. Atente para as curvaturas das linhas.

Solução:

```
library(sf)
library(rnaturalearth)

br_valid_poly <- sf::st_transform(br_valid, crs = 5880)
plot(br_valid_poly$geometry, col = "gray", axes = TRUE, graticule = TRUE)
```



**15.5** Utilizando a função **st\_centroid** do pacote **sf**, crie um vetor chamado

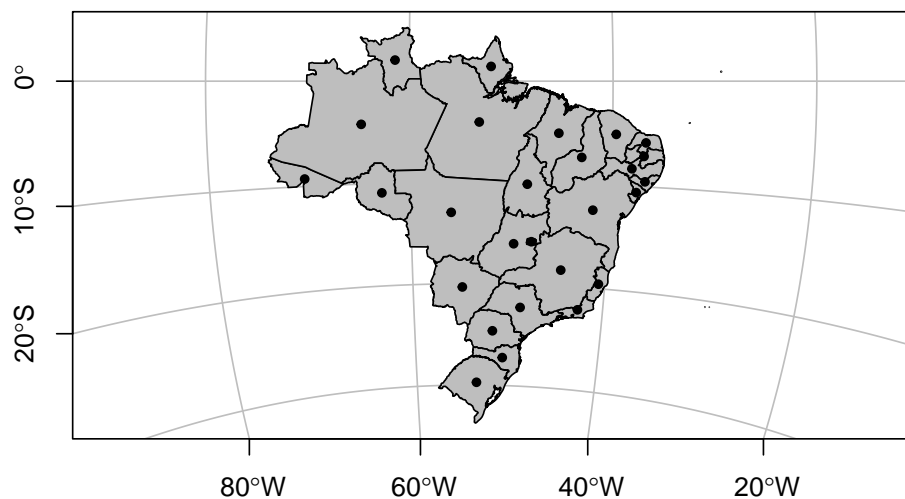
**br\_valid\_cen** que armazenará o centroide de cada estado brasileiro do objeto **br\_valid** do exercício 2 e plot o resultado.

Solução:

```
library(sf)
library(rnaturalearth)

br_valid_poly_cen <- sf::st_centroid(br_valid_poly)

plot(br_valid_poly$geometry, col = "gray", axes = TRUE, graticule = TRUE)
plot(br_valid_poly_cen$geometry, pch = 20, add = TRUE)
```



**15.6** Ajuste o limite e máscara do objeto raster criado no exercício 3 para o limite do Brasil, atribuindo ao objeto **ra\_br**. Depois reprojete esse raster para a mesma projeção utilizada no exercício 4 com o nome **ra\_br\_poly** e plote o mapa resultante.

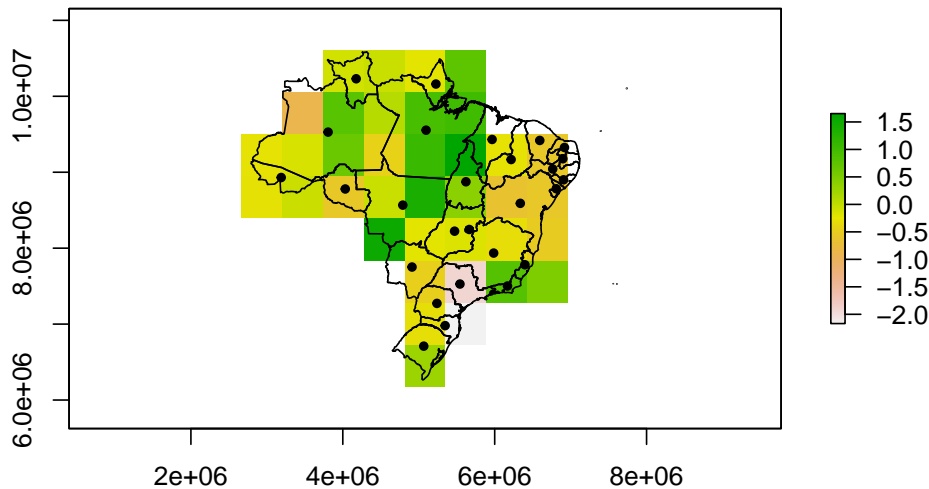
Solução:

```
library(raster)

ra_br <- ra %>%
  raster::crop(br_valid) %>%
  raster::mask(br_valid)

ra_br_poly <- raster::projectRaster(ra_br, crs = "+init=epsg:5880")

plot(ra_br_poly)
plot(br_valid_poly$geometry, add = TRUE)
plot(br_valid_poly_cen$geometry, pch = 20, add = TRUE)
```



**15.7** Extraia os valores de cada pixel do raster criado no exercício 6 para os centroides dos estados do Brasil criado no exercício 5, atribuindo à coluna **val** do objeto espacial chamado **br\_valid\_poly\_cent\_ra**.

Solução:

```
br_valid_poly_cent_ra <- br_valid_poly_cen %>%
  dplyr::mutate(val = raster::extract(ra_br_poly, .))
head(br_valid_poly_cent_ra$val)
#> [1] 0.2676713 -0.1098317 0.9373188 -0.2846465 -0.2323340
#> [6] -0.4757399
```

**15.8** Crie um mapa final usando os resultados dos exercícios 4, 5 e 6. Utilize o pacote **tmap** e inclua todos os principais elementos de um mapa.

Solução:

```
library(tmap)

tm_shape(ra_br_poly) +
  tm_raster(title = "Raster") +
  tm_shape(br_valid_poly) +
  tm_borders() +
  tm_shape(br_valid_poly_cent_ra) +
  tm_bubbles(col = "val", size = .2, legend.col.show = FALSE) +
  tm_graticules(lines = FALSE,
    labels.format = list(big.mark = ""),
    labels.rot = c(0, 90),
    labels.size = .7) +
  tm_compass(position = c("right", "top"), size = 2) +
  tm_scale_bar(size = 1) +
  tm_xlab("Longitude", size = 1) +
```

```
tm_ylab("Latitude", size = 1) +
tm_credits("CRS: SIRGAS2000/Polícônica", position = c(.6, .15), size = .6) +
tm_credits("Fonte: Natural Earth (2022)", position = c(.6, .12), size = .6) +
tm_layout(main.title = "Estados do Brasil",
           main.title.position = c(.1, .95),
           main.title.size = 1.5,
           title.fontface = "bold",
           legend.position = c("left", "bottom"),
           legend.title.fontface = "bold")
```

