## Lista Prática 1

**Exercício 1** (`swirl`). *O pacote* `swirl` *permite aprender a usar o R de forma interativa diretamente no console do R. Para instalar e inicializar o* `swirl`*, use os comandos*[1]*:*

```
install.packages("swirl")
library(swirl)
swirl()
```

*Aparecerá uma mensagem de inicialização e, então, digite seu nome diretamente no console. Se esta for a primeira vez utilizando o* `swirl` *aparecerá algumas instruções e, depois, será mostrado o seguinte repositório de cursos:*

```
1: R Programming: The basics of programming in R
2: Regression Models: The basics of regression modeling in R
3: Statistical Inference: The basics of statistical inference in R
4: Exploratory Data Analysis: The basics of exploring data in R
5: Don't install anything for me. I'll do it myself.
```

*Neste exercício, utilizaremos o curso "R Programming", portanto, digite 1 para instalá-lo e, logo, digite 1 novamente para acessá-lo. Então, para iniciar uma lição, é necessário escolher uma na seguinte lista:*

```
 1: Basic Building Blocks      2: Workspace and Files
 3: Sequences of Numbers       4: Vectors
 5: Missing Values             6: Subsetting Vectors
 7: Matrices and Data Frames   8: Logic
 9: Functions                 10: lapply and sapply
11: vapply and tapply         12: Looking at Data
13: Simulation                14: Dates and Times
15: Base Graphics
```

*Faça as seguintes lições no* `swirl`*:*

*a) 1: Basic Building Blocks*

*b) 3: Sequences of Numbers*

*c) 4: Vectors*

*d) 5: Missing Values*

*e) 6: Subsetting Vectors*

*f) 7: Matrices and Data Frames*

*g) 8: Logic*

*h) 13: Simulation*

[1] Caso tenha dificuldades, assistir vídeo: `https://youtu.be/olOJfAjzdO8`

**Note que, para atribuição de valores a objetos, o* `swirl()` *exige que isso seja feito via "<-", ou seja, não aceita respostas com atribuição por "=").*

**Exercício 2.** *Considere os dois vetores* `x = c(1, 3, 5:8)` *e* `y = c(3, 2:-1, 10)`*.*

*a) O que é produzido pela expressão* `rbind(x, y)`*? Qual é a sua classe?*

*b) Crie um vetor* `z` *com os elementos dos vetores* `x` *e* `y`*. Usando uma função e o vetor* `z` *como input dela, crie o mesmo objeto produzido no item (a).*

*Resposta*:

a) Produz uma matriz com 2 linhas e 6 colunas:

```
  [,1] [,2] [,3] [,4] [,5] [,6]
x    1    3    5    6    7    8
y    3    2    1    0   -1   10
```

b)
```
z = c(x, y)
matrix(z, nrow=2, ncol=6, byrow=TRUE)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    3    5    6    7    8
[2,]    3    2    1    0   -1   10
```

□

**Exercício 3.** *Suponha uma lista definida por* `x = list(2, "a", "b", TRUE)`*.*

*a) O que* `x[[2]]` *retorna? Qual é a sua classe?*

*b) E* `x[[1]]`*?*

*c) E* `x[2]`*?*

*Resposta*:

a) Retorna o vetor texto (*character*) "a"

b) Retorna o vetor numérico (*numeric*) 2

c) Retorna a lista composta pelo vetor texto (*character*) "a".

Note que o operador `[]` retorna um objeto da mesma classe do objeto originário (aqui uma lista), enquanto `[[]]` retorna um objeto na classe do vetor que compõe a lista. □

**Exercício 4.** *Suponha o vetor* `x = c(3, 5, 1, 10, 12, 6)`*. Escreva o vetor x resultante após executar os comandos abaixo. Lembre-se de atribuir o vetor acima ao objeto* `x` *antes de iniciar cada item deste exercício.*

*a)* `x[x == 6] = 0`

*b)* `x[x != 12] = 0`

*c)* `x[x < 6 & x > 1] = 0`

*d)* `x[x %in% 1:5] = 0`

*e)* `x[!(x %in% 5:10)] = 0`

*Resposta*:

a) $\{3, 5, 1, 10, 12, 0\}$

b) $\{0, 0, 0, 0, 12, 0\}$

c) $\{0, 0, 1, 10, 12, 6\}$

d) $\{0, 0, 0, 10, 12, 6\}$

e) $\{0, 5, 0, 10, 0, 6\}$

□

**Exercício 5.** *Usando a base de dados* `airquality`[2]*, responda os seguintes itens:*

*a) Valor de 'Ozone' na linha 47*

*b) Quantidade de missing values (NA) na coluna 'Ozone'*

*c) Média da 'Ozone'*

*d) Média de 'Temp' quando 'Month' = 6*

*e) Média de 'Solar.R' quando 'Ozone' > 31 e 'Temp' > 90*

*f) Maior valor de 'Ozone' quando 'Month' = 5*

*Resposta*:

a) `airquality$Ozone[47]` ou `airquality[47, 1]` $\rightsquigarrow$ Resposta: 21

b) Para criar um vetor de TRUE/FALSE condicionada ao elemento ser um missing value, usamos `is.na()`. Lembre-se também que TRUE e FALSE correspondem, respectivamente aos números 1 e 0. Portanto, ao somarmos um vetor de TRUE/FALSE, obtemos a quantidade total de TRUE's.
`sum( is.na(airquality$Ozone) )` $\rightsquigarrow$ Resposta: 37

c) Do item anterior, sabemos que Ozone possui missing values e, portanto, ao utilizarmos a função `mean()` sem remover os missing values, o R reporta um NA.
`mean(airquality$Ozone, na.rm = TRUE)` $\rightsquigarrow$ Resposta: 42.12931

[2]Base de dados nativa do R, basta escrever o nome dela para acessá-la.

d) `mean(airquality$Temp[airquality$Month == 6], na.rm = TRUE)` ⇝ Resposta: 79.1

e) `mean( airquality$Solar.R[ airquality$Ozone > 31 & airquality$Temp > 90 ], na.rm = TRUE )` ⇝ Resposta: 212.8

f) `max(airquality$Ozone[airquality$Month == 5], na.rm = TRUE)` ⇝ Resposta: 115

□

**Exercício 6.** *Considere o vetor de números inteiros `1:100`. Faça uma reamostragem de 100 números sem reposição deste vetor. Reordene este vetor de forma decrescente usando estruturas condicionais e de repetição/loop.*

*Resposta*: Primeiro, crie o vetor descrito no enunciado:

```
vetor = sample(1:100, 100)
print(vetor)
```

```
  [1]  80   9  20  95  24  43  65  72  29  88  47  71  56  69  37  54  79
 [18]  16   3  84  67  12  28  83   6  50  25  86  70  30  22 100  36  61
 [35]  51  45  10  98  14  26  89  55  48  21  44  66  64  94  15  49  40
 [52]  96  97   8   5  53  68  35  18  34  52  32  62   7  60  58  93  87
 [69]  38  77  82  46  23   4  27  31  11  19  99  13  91   2   1  81  57
 [86]  85  17  63  41  76  59  74  42  90  39  78  92  73  33  75
```

Iniciando no 1º elemento do vetor, faremos uma comparação entre este elemento e o seguinte. Se este for maior do que aquele, inverteremos a ordem e, caso o elemento seja seguido por um número menor, a ordem continuará. Logo, faremos a mesma análise para a próxima dupla do vetor e repetiremos até chegar ao penúltimo elemento do vetor, que será comparado ao último.

```
for (i in 1:(length(vetor) - 1))
        if (vetor[i] < vetor[i + 1]) {
            aux = vetor[i + 1]
            vetor[i + 1] = vetor[i]
            vetor[i] = aux
        }
    }
```

```
  [1]  80  20  95  24  43  65  72  29  88  47  71  56  69  37  54  79  16
 [18]   9  84  67  12  28  83   6  50  25  86  70  30  22 100  36  61  51
 [35]  45  10  98  14  26  89  55  48  21  44  66  64  94  15  49  40  96
 [52]  97   8   5  53  68  35  18  34  52  32  62   7  60  58  93  87  38
 [69]  77  82  46  23   4  27  31  11  19  99  13  91   3   2  81  57  85
 [86]  17  63  41  76  59  74  42  90  39  78  92  73  33  75   1
```

Note que este loop fez apenas o menor número ir à última posição do vetor. Precisamos rodar um outro loop para que o 2º menor número vá para a penúltima posição e assim por diante. Note que o loop seguinte não precisa ser feito até o último elemento do vetor, já que ele já está na posição correta. Então, a cada vez que rodamos um loop, precisamos fazer uma comparação a menos.

```
for (num_loops in 0:98) {
    for (i in 1:(length(vetor) - 1 - num_loops)) {
        if (vetor[i] < vetor[i + 1]) {
```

```
            aux = vetor[i + 1]
            vetor[i + 1] = vetor[i]
            vetor[i] = aux
        }
    }
}
```

```
  [1] 100  99  98  97  96  95  94  93  92  91  90  89  88  87  86  85  84
 [18]  83  82  81  80  79  78  77  76  75  74  73  72  71  70  69  68  67
 [35]  66  65  64  63  62  61  60  59  58  57  56  55  54  53  52  51  50
 [52]  49  48  47  46  45  44  43  42  41  40  39  38  37  36  35  34  33
 [69]  32  31  30  29  28  27  26  25  24  23  22  21  20  19  18  17  16
 [86]  15  14  13  12  11  10   9   8   7   6   5   4   3   2   1
```

□

**Exercício 7.** *Considere* `x = 1`*. Crie um loop que, a cada repetição, divide o valor de* `x` *pela metade. Faça isso até* `x < 1e-100` *($1 \times 10^{-100}$). Quantas repetições são necessárias para que* `x` *atinja esse valor?*

*Resposta*: 333 loops

```
x = 1
contador = 0

while (x > 1e-100) {
    x = x/2
    contador = contador + 1
}
```

□

**Exercício 8.** *Considere que João precise definir um* `x` *considerando uma função de desutilidade (loss function) dada por* $y = x^4 - 6x^2 + 4x + 4$.

a) *Suponha que João só possa escolher* $0 \leq x \leq 3$*. Como há infinitos pontos neste intervalo (e o computador não consegue tratar cálculos com infinitos valores), discretizaremos os possíveis valores de* $x$ *criando um vetor* `x = seq(from = 0, to = 3, length = 61)`*. Qual é o valor de* $x$ *que minimiza a desutilidade de João,* $y$*?*

b) *Suponha* $-3 \leq x \leq 3$ *e use* `x = seq(from = -3, to = 3, length = 121)`*. O valor de* $x$ *que minimiza* $y$ *permanece o mesmo?*

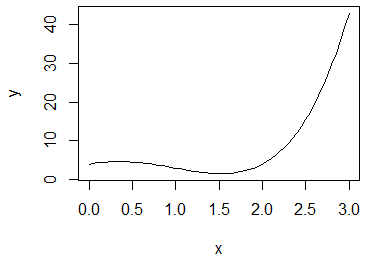
c) *Agora, suponha* $-100 \leq x \leq 100$ *e use* `x = seq(from = -100, to = 100, length = 121)`*. O que acontece?*

**Dica: Compare os casos usando gráficos:* `plot(x, y, type="l")`*.*

*Resposta*:

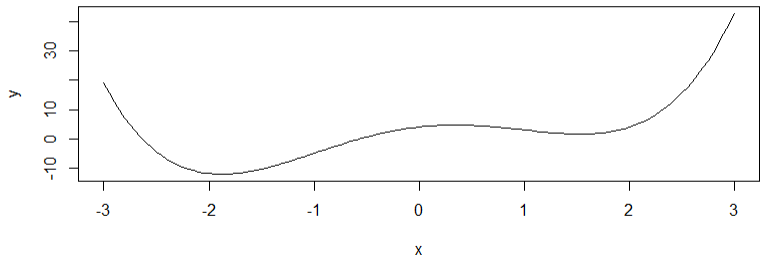
a) Resposta: 1,55

```
x = seq(from = 0, to = 3, length = 61)
y = x^4 - 6*x^2 + 4*x + 4
x[which.min(y)]
```



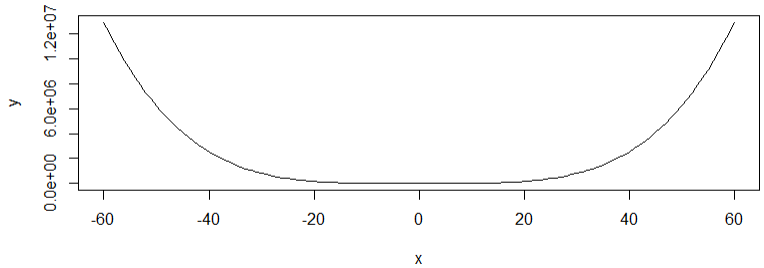

b) Resposta: -1,9 $\rightsquigarrow$ valor que minimiza $f(x)$ mudou, pois o intervalo $0 \leq x \leq 3$ selecionava um mínimo local.

```
x = seq(from = -3, to = 3, length = 121)
y = x^4 - 6*x^2 + 4*x + 4
x[which.min(y)]
```




c) Resposta: -2 $\rightsquigarrow$ aumentar o intervalo de potenciais valores de $x$, sem aumentar proporcionalmente a quantidade de pontos, diminuiu a precisão de seleção do $x$ que minimiza $f(x)$. O mesmo ocorreria se diminuísse a quantidade de pontos dentro do intervalo.

```
x = seq(from = -60, to = 60, length = 121)
y = x^4 - 6*x^2 + 4*x + 4
x[which.min(y)]
```




□