Insper EDUCAÇÃO EXECUTIVA ONLINE

Live Learning AO VIVO

Programação para Não Programadores

Aula 1

Prof. Magno Severino e Prof. Marina Muradian

13/04/2021

Objetivos de aprendizagem

- Conhecer os painéis da área de trabalho do RStudio.
- Realizar operações matemáticas utilizando `R`.
- Diferenciar os tipos de dados que podem ser armazenados na linguagem `R`.

Personalizando os painéis do RStudio

Para personalizar a aparência e a posição dos painéis do RStudio, acesse o menu `Tools` e escolha a opção `Global Options...`. Na aba `Apearance` você pode configurar a fonte e a cor do plano de fundo. Na aba `Pane Layout` você pode configurar a disposição dos painéis na tela de trabalho.

Utilizando o `R` como uma calculadora

O operador `+` realiza a adição entre dois elementos.

```
1 + 2
```

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

Um vetor é um conjunto ordenado de valores. O operador `:` cria uma sequência a partir de um número até outro. A função `c` concatena valores, criando um vetor.

```
1:5
```

```
## [1] 1 2 3 4 5
```

```
c(1, 2, 3, 4, 5)
```

```
## [1] 1 2 3 4 5
```

Além de adicionar dois números, o operador `+` pode ser usado para adicionar dois vetores.

```
1:5 + 6:10
```

```
## [1]  7  9 11 13 15
```

```
c(1, 2, 3, 4, 5) + c(6, 7, 8, 9, 10)
```

```
## [1]  7  9 11 13 15
```

```
1:5 + c(6, 7, 8, 9, 10)
```

```
## [1]  7  9 11 13 15
```

Os próximos exemplos mostram subtração, multiplicação, exponenciação e divisão.

```
c(2, 3, 5, 7, 11, 13) - 2       #subtração
```

```
## [1]  0  1  3  5  9 11
```

```
-2:2 * -2:2                     #multiplicação
```

```
## [1] 4 1 0 1 4
```

```
(1:10) ^ 2                        #exponenciação
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

```
1:10 / 3                        #divisão
```

```
##  [1] 0.3333333 0.6666667 1.0000000 1.3333333 1.6666667 2.0000000 2.3333333
##  [8] 2.6666667 3.0000000 3.3333333
```

Operadores Lógicos

- Para verificar a igualdade entre dois valores, usamos `==` (note que são dois sinais de igual). Este operador pode ser usado também para vetores.
- Para checar se dois valores são diferentes, usamos o operador `!=`
- Maior que e menor que, respectivamente, `>` e `<` (ou `>=` e `<=` quando igualdade é permitida).

Veja alguns exemplos abaixo:

```
c(3, 4 - 1, 1 + 1 + 1) == c(3, 3, 3)
```

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

```
c(3, 4 - 1, 1 + 1 + 1) == 3
```

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

```
1:3 != 3:1
```

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

```
exp(1:5) < 100
```

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

O operador == pode ser usado também para comparar cadeias de caracteres. Aqui, letras maiúsculas e minúsculas são consideradas diferentes

```
c("A", "B", "C") == c("A", "b", "d")
```

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

Atribuindo variáveis

Fazer cálculos com o R é bem simples e útil.

Na maior parte das vezes, queremos armazenar os resultados para uso posterior.

Assim, podemos atribuir valor à uma variável, através do operador `<-`

```
a <- 1

b <- 5 * 3

x <- 1:5

y <- 6:10
```

Agora, podemos reutilizar esses valores para fazer outros cálculos.

```
a + 2 * b
```

```
## [1] 31
```

```
x + 2 * y - 3
```

```
## [1] 10 13 16 19 22
```

Observe que não temos que dizer ao R qual a estrutura da variável, se é um número (as variáveis `a` e `b`) ou vetor (`x` e `y`).

Números especiais

Para facilitar operações aritméticas, R suporta quatro valores especiais de números: `Inf`, `-Inf`, `NaN` e `NA`.

Os dois primeiros representam infinito positivo e negativo. `NaN` é um acrônimo inglês para "not a number", ou seja, não é um número.

Ele aparece quando um cálculo não faz sentido, ou não está definido. `NA` significa "not available", ou seja, não disponível, e representa um valor faltante.

```
c(Inf + 1, Inf - 1, Inf - Inf, NA + 1)
```

```
## [1] Inf Inf NaN  NA
```

```
c(0 / 0, Inf / Inf, 1 / Inf)
```

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

Além de números, podemos utilizar valores lógicos (verdadeiro e falso).

```
1:10 >= 5
```

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

```r
z <- 1:10 >= 5   #armazena o resultado na variavel z

w <- 1:10 <= 3   #armazena o resultado na variavel w

y <- 1:10 <= 7   #armazena o resultado na variavel y
```

Operações lógicas podem ser feitas através dos operadores *negação* (!), *e* (&) e *ou* (|), além de combinações entre elas

```r
!z
```

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

```r
z & w
```

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

```r
z | w
```

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

```r
(z & y) | w
```

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

Analisando variáveis

Agora, vamos examinar as propriedades das variáveis que definimos anteriormente.

```r
x
```

```
## [1] 1 2 3 4 5
```

```r
z
```

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

Classes

À toda variável no R é atribuida uma classe. Para descobrir a classe de uma variável, use a função `class(nome_da_variavel)`

```r
class(x)
```

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

```
class(z)
```

```
## [1] "logical"
```

Note que `class` é uma função. Para obter ajuda sobre uma função, utilize o comando `?nome_da_funcao`. Exemplo:

```
?class
```

Além das classes numérica e lógica, existem duas outras importantes: `character` para armazenamento de texto e `factor` para armazenamento de dados categóricos. No próximo exemplo, criamos um vetor de caracteres usando o operador `c` (o mesmo que foi usado para criar vetores numéricos):

```
palavras <- c("Eu", "estou", "aprendendo", "a", "programar", "em", "r")
class(palavras)
```

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

Um exemplo de dado categórico é o gênero de uma pessoa.

```
genero <- factor(c("F", "M", "M", "F", NA, "F"))

genero
```

```
## [1] F    M    M    F    <NA> F
## Levels: F M
```

```
class(genero)
```

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

```
levels(genero)    #mostra as categorias do fator
```

```
## [1] "F" "M"
```

Verificando e alterando classes

Para verificar se uma variável pertence à uma classe, utilize a função `is.*`, substituindo o `*` pelo nome da classe. Veja os exemplos:

```
is.character("Eu estou aprendendo a programar em R")
```

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

```
is.logical(FALSE)
```

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

```
is.numeric(1)
```

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

As vezes, é necessário mudar o tipo de uma variável, para isso, usamos a função `as(variavel, tipo)` ou `as.*`. *Cuidado*! Em alguns casos essa operação pode gerar resultados indesejáveis.

```
x <- "123.456"

class(x)
```

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

```
as(x, "numeric")
```

```
## [1] 123.456
```

```
as.numeric(x)
```

```
## [1] 123.456
```

```
as.numeric("abc")
```

```
## Warning: NAs introduzidos por coerção
```

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

Examinando variáveis

Além de verificar o valor de uma variável, muitas vezes é útil ver algum tipo de resumo do objeto. A função `summary` faz isso, dando informações apropriadas de acordo com os diferentes tipos de dados. O resumo de variáveis numéricas contém média, mediana e alguns quantis.

```
numeros <- runif(10)
# a funcao runif amostra 10 números aleatórios
# uniformemente distribuidos entre 0 e 1.
summary(numeros)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.07054 0.18158 0.25302 0.40596 0.64972 0.99868
```

Variáveis categóricas e fatores são resumidos pelo número de elementos em cada vetor.

```
set.seed(123)
# A função set. seed serve para se poder reproduzir os resultados
# dos geradores de números pseudo-aleatórios

letras <- sample(c("a", "e", "i", "o", "u"), 10, replace = TRUE)
```

```r
# a função sample amostra de maneira aleatória as vogais,
# com reposição para obter 10 vogais

letras <- as.factor(letras)

summary(letras)
```

```
## a e i o u
## 1 3 4 1 1
```

```r
bool <- sample(c(TRUE, FALSE, NA), 30, replace = TRUE)

summary(bool)
```

```
##    Mode   FALSE    TRUE    NA's
## logical       6      12      12
```

O Workspace

Enquanto estiver trabalhando o R, pode ser útil saber o nome de todas as variáveis que você criou. Para isso, utilize a função `ls`.

```r
ls()
```

```
##  [1] "a"        "b"        "bool"     "genero"   "letras"   "numeros"
##  [7] "palavras" "w"        "x"        "y"        "z"
```

Após terminar as operações com uma variável que não será mais utilizada, podemos removê-la usando a função `rm`

```r
rm(letras)

rm(bool)

rm(list = ls()) #remove todas as variáveis. USE COM CUIDADO!
```

Pratique seu conhecimento!

Exercício 1

a) Armazene os números de 1 a 1000 na variável `x`.

b) Calcule o quadrado dos valores de `x` e armazene-os em `y`.

c) Selecione aleatoriamente 100 elementos de `y` e armazene em `z`

d) Mostre os dados contidos em `z`.

e) Obtenha um sumário de `z`.

f) Conte quantos elementos são maiores que 5000.

Exercício 2

Leia `?which.min` e `?which.max`. Encontre os índices das entradas com os valores mínimo e máximo do vetor `numeros` definido abaixo.

```
set.seed(1234)

numeros <- sample(1:50, size=10)

numeros
```

Exercício 3 (desafio)

Execute o código abaixo.

```
set.seed(123)

x <- sample(letters, size = 15, replace = TRUE)

x
```

Leia `?letters` e `?which`. O operador `%in%` devolve um vetor de valores lógicos indicando as ocorrências dos elementos de um vetor em outro vetor. Usando o operador de `%in%` e a função `which`:

a) Encontre todas as ocorrências de vogais no vetor `x`.

b) Encontre os índices das entradas do vetor `x` em que temos consoantes.