# Aula 31/10

## Tipos de dados

# Constantes

# - Numérica (double): 1, 10, 1.6, 0.0908

# - Numérica inteira: 1L, 10L

# - Numérica complexa: 8i, 10i + 10L

# - Lógica ou booleana: TRUE, FALSE

# Alfanumérica/caractere/string: "A", "BiCho do MaTo", "“Earl J. Waggedorn“", "!˜#$%ˆ&\*()\_+| ’:;<>?/,.˜`"

# Outras constantes:

# NULL: NULL representa o **objeto nulo** em R: é uma *palavra reservada*. NULL geralmente é retornado por expressões e funções cujo valor é indefinido.

# NaN: NaN significa "**Não é um número**". (Isso se aplica a valores numéricos e partes reais e imaginárias de valores complexos). NaN é uma palavra reservada na linguagem R.

# InF : Inf e -Inf definem que os elementos numéricos/com plexos são infinitos positivo e negativo

# ------------

# Variável

# Uma definição: Variáveis são **contêineres** (recipientes) para armazenar valores de dados.

# A linguagem R não tem um comando para **declarar** uma variável. Uma variável é criada no momento em que você atribui um valor a ela. Para atribuir um valor a uma variável, utiliza-se o sinal **<-** ou **=** .

Real <- 1

Inteiro <- 10L

Complexa = 9i

Booleana = TRUE

Caractere <- '@ eu sou !) uma cadeia # ou 1ª. seqüência $%&?><M de caracteres: letras aeiou, números 123456789 e quetais :?/+\_\_0'

# Para mostrar (ou imprimir) o valor (ou conteúdo) de uma variável, basta digitar o nome desta variável

Inteiro

Caractere

# Outros exemplos de variáveis:

nome.do.aluno = 'Maria das Dores'

nome\_do\_aluno = 'AustregésilodeAthayde'

NOME-DO-ALUNO <- "Mumú das Candongas" # Error in NOME - DO - ALUNO <- object 'NOME' not found

idade <- 40

NOMEDOALUNO = "Mumú das Candongas"

NOME\_DO\_ALUNO = "Mafuá do Malungo"

str <- "Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."

# Vetores

# Um vetor é simplesmente **uma lista de itens** que são do mesmo tipo.

# Para combinar a lista de itens em um vetor, no R utiliza-se **a função c()** e coloca-se os itens separados por vírgula.

# Observação:

# A função c() faz parte da *biblioteca básica* do R. O método padrão **combina** seus argumentos **para formar um vetor**. Todos os argumentos são forçados a um tipo de dados comum, que é o tipo do valor que a função retorna, e todos os atributos, exceto os nomes, são removidos.

# Exemplo: variável vetorial chamada frutas, que combina caracteres:

# Vetor de strings

frutas = c('pitanga', "banana", "pêssego", "amora", "mamão", 'goiaba', "maçã", 'acerola', "laranja", "manga", 'abacaxi', 'poncã', 'melancia', "romã", 'limão', 'jaca', 'abacate', "açaí", 'melão', 'tangerina')

# Imprimir frutas

frutas

# Neste exemplo, criamos um vetor que combina valores numéricos:

# Vetor de valores numéricos

números <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 0)

# Imprimir números

números

# Vetor de valores lógicos

log\_values <- c(T, F, T, F)

# Imprimir os valores lógicos

log\_values

# Tamanho de um vetor

# Para descobrir quantos itens um vetor possui, utiliza-se a função **length():**

# Exemplo

frutas = c('pitanga', "banana", "pêssego", "amora", "mamão", 'goiaba', "maçã", 'acerola', "laranja", "manga", 'abacaxi', 'poncã', 'melancia', "romã", 'limão', 'jaca', 'abacate', "açaí", 'melão', 'tangerina')

length (frutas)

números <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 0)

length(números)

# Classificar/ordenar um vetor

# Para classificar os itens de um vetor, alfabeticamente ou numericamente, utiliza-se a função sort():

# Exemplo

frutas

sort(frutas) # classifica a string

números

sort(números) # ordena os números

# Acessar os valores de um vetore

# Pode-se acessar os itens do vetor referindo-se ao seu **número de índice** *entre colchetes* [].

# O primeiro item tem índice 1, o segundo item tem índice 2 e assim por diante:

# Exemplo:

# Acessar/obter o primeiro item (pitanga)

frutas[1]

# Obter os quatro primeiros itens do vetor frutas

frutas[1:4]

# Também pode-se utilizar números negativos de índice para acessar todos os itens, exceto os especificados:

números[-4]

números[-4:0]

frutas[-3]

frutas[-3:0]

frutas[0:-4]

# Pode-se também acessar vários elementos referindo-se a diferentes posições de índice com a função c().

# Exemplo: Obter o primeiro e o terceiro itens (pitanga e pêssego)

> frutas[c(1, 3)]

# Exemplo: Obter o quarto e do nono ao décimo primeiro itens (amora, laranja, manga, abacaxi)

frutas[c(4, 9:11)]

# Exemplo: Obter o primeiro e o quarto itens, do sexto ao oitavo itens e o décimo elemento (1, 4, 6, 7, 8, 0)

números[c(1, 4, 6:8,10)]

# [1] 1 4 6 7 8 0

# Alterar um item

# Para alterar o valor de um item específico, deve-se referenciar o número do índice e atribuir um valor para aquela posição.

# Exemplo:

frutas

# [1] "pitanga" "banana" "pêssego" "amora" "mamão" "goiaba" "maçã" "acerola"

# [9] "laranja" "manga" "abacaxi" "poncã" "melancia" "romã" "limão" "jaca"

# [17] "abacate" "açaí" "melão" "tangerina"

# Trocando "banana" por "pêra"

frutas[2] <- "pera"

# Imprimir frutas

frutas

# [1] "pitanga" "pera" "pêssego" "amora" "mamão" "goiaba" "maçã" "acerola"

# [9] "laranja" "manga" "abacaxi" "poncã" "melancia" "romã" "limão" "jaca"

#[17] "abacate" "açaí" "melão" "tangerina"

# Matrizes

# Uma matriz é um conjunto de dados bidimensional com colunas e linhas.

# Uma coluna é uma representação vertical de dados, enquanto uma linha é uma representação horizontal de dados.

# Uma matriz pode ser criada com a função matrix(). São especificados os parâmetros nrow e ncol para obter a quantidade de linhas e colunas:

# Exemplo

# Criando uma matriz

esta.matriz <- matrix(c(1,2,3,4,5,6), nrow = 3, ncol = 2, byrow = FALSE)

# Imprime a matriz

esta.matriz

# Pode-se também criar uma matriz com strings:

# Exemplo

esta.matrizs <- matrix(c("maçã", "banana", "cereja", "laranja"), nrow = 2, ncol = 2)

# Imprime a matriz

esta.matrizs

# Acessar Itens da Matriz

# Você pode acessar os itens usando colchetes [ ].

# O primeiro número (1) entre colchetes especifica a posição da linha, enquanto o segundo número (2) especifica a posição da coluna:

# Exemplo

esta.matrizs[1, 2]

# [1] "cereja"

# Uma linha inteira pode ser acessada se for colocada uma vírgula após o número entre colchetes:

#Exemplo

esta.matriz[2,]

# A coluna inteira pode ser acessada se você especificar uma vírgula antes do número entre colchetes:

# Exemplo

esta.matriz[,2]

# [1] 4 5 6

esta.matrizs[,2]

# [1] "cereja" "laranja"

# Acessar mais de uma linha

# Mais de uma linha pode ser acessada usando a função c():

# Exemplo

outra\_matrix <- matrix(c("maçã", "banana", "cereja", "laranja", "uva", "abacaxi", "pêra", "melão", "figo"), nrow = 3, ncol = 3)

outra\_matrix

# [,1] [,2] [,3]

# [1,] "maçã" "laranja" "pêra"

# [2,] "banana" "uva" "melão"

# [3,] "cereja" "abacaxi" "figo"

outra\_matrix[c(1,2),]

# [,1] [,2] [,3]

# [1,] "maçã" "laranja" "pêra"

# [2,] "banana" "uva" "melão"

>

# Acessar mais de uma coluna

# Mais de uma coluna pode ser acessada usando a função c():

# Exemplo

outra\_matrix[, c(1,2)]

# [,1] [,2]

# [1,] "maçã" "laranja"

# [2,] "banana" "uva"

# [3,] "cereja" "abacaxi"

# Verificar se existe um item

# Para descobrir se um item especificado está presente em uma matriz, deve-se utilizar o operador %in%:

# Exemplo

# Verifique se "maçã" está presente na matriz:

"maçã" %in% outra\_matrix

# [1] TRUE

"melancia" %in% outra\_matrix

# [1] FALSE

# Número de linhas e colunas de uma matriz

# A função dim() mostra o número de linhas e colunas em uma matriz

# Exemplo

dim(outra\_matrix)

# [1] 3 3

# Comprimento da Matriz

# A função length() mostra a dimensão de uma matriz

# Exemplo

length(outra\_matrix)

# [1] 9

# Data Frames (Quadros de Dados)

# Data Frames são dados armazenados e exibidos em formato de tabela.

# Os Data Frames podem armazenar/utilizar diferentes tipos de dados.

# A primeira coluna pode ser do tipo caractere, a segunda pode ser numéricas e a terceira pode conter dados lógicos.

# Entretanto, cada coluna deve ter o mesmo tipo de dados.

# Pode-se utilizar a função data.frame() para criar um quadro de dados:

# Exemplo

# Cria um dataframe

Data\_Frame <- data.frame (

Treinamento = c("Força", "Resistência", "Outro"),

Pulso = c(100, 150, 120),

Duração = c(60, 30, 45)

)

# Mostrando o data frame

Data\_Frame

# Treinamento Pulso Duração

# 1 Força 100 60

# 2 Resistência 150 30

# 3 Outro 120 45

# Um data frame **é semelhante a uma matriz mas as suas colunas têm nomes e podem conter dados de tipo diferente**.

# Um data frame é a maneira mais comum de armazenar dados em R e, geralmente, é a estrutura de dados mais usada para análises de dados.

# Em resumo, um data frame é **uma lista de vetores de igual comprimento**. **Cada elemento** da lista pode ser pensado como **uma coluna** e o tamanho de cada elemento da lista é **o número de linhas**.

# Por isso, um data frame pode armazenar diferentes classes de objetos em cada coluna (ou seja, numérico, caractere, fator, lógico).

# Acessar itens de um data frame

# Podemos usar colchetes simples [ ], colchetes duplos [[ ]] ou $ para acessar as colunas de um data frame.

# Exemplos

Data\_Frame[3]

Data\_Frame[["Duração"]]

Data\_Frame$Duração

# Para acessar o conteúdo das colunas de uma linha, podemos usar colchetes simples [ ], o índice (número) da linha e vírgula.

# Exemplo:

Data\_Frame[3,]

Treinamento Pulso Duração

3 Outro 120 45

# Para acessar um valor (conteúdo) específico, é preciso indicar, entre colchetes [], o número da linha e o número (ou o nome) da coluna, similar à matriz.

# Exemplo

Data\_Frame [1, 1]

Data\_Frame [1, "Treinamento"]

# Quantidade de linhas e colunas

# Para encontrar a quantidade de linhas e colunas em um Data Frame, utiliza-se a função dim().

# Exemplo

dim(Data\_Frame)

# [1] 3 3

# Para encontrar o número de colunas, pode-se utilizar a função ncol() e para encontrar o número de linhas, nrow().

# Exemplo

ncol(Data\_Frame)

[1] 3

nrow(Data\_Frame)

[1] 3

# Para descobrir o número de colunas em um Data Frame, pode-se utilizar a função length() (semelhante a ncol()).

# Exemplo

length(Data\_Frame)

[1] 3

# Outra forma de se utilizar um data frame em R é por meio da importação dos dados.

# Existem várias formas de “trazer de fora” os dados, dependendo do tipo e da estrutura do arquivo a ser importado.

# O processo mais comum é utilizar uma função read...().

# Um arquivo do tipo Excel (com extensão .xls ou .xlsx) para ser importado necessita do “carregamento” de uma biblioteca (ou “package”, em R) adicional (add-on).

# Neste exemplo, utilizamos a biblioteca readxl e sua função read\_excel().

# Para conhecer melhor a estrutura e as funções que compõem uma biblioteca, deve-se digitar, na Console, ?nome\_da\_biblioteca (por exemplo: ? readxl). O mesmo para verificar o conteúdo e os parâmetros da função. Por exemplo: ?read\_excel.

# Exemplo: importar os dados da planilha “exercicio1.xls”.

df\_1 <- read\_excel("./dados/exercicio1.xls")

# “Ver” o conteúdo do data frame df\_1

View(df\_1)

# Quantidade de linhas e colunas

dim(df\_1)

# [1] 10 1

# Número de colunas ncol() e número de linhas nrow().

ncol(df\_1)

[1] 1

nrow(df\_1)

[1] 10

# Acessar colunas de um data frame

df\_1[1]

df\_1[['Taxas de juros']]

df\_1$'Taxas de juros'

# Algumas operações em R

# Cálculo da média aritmética

mean(df\_1$'Taxas de juros')

# [1] 2.595

median(df\_1$'Taxas de juros')

[1] 2.605

# Desvio Padrão

sd(df\_1$'Taxas de juros')

[1] 0.04453463

# Variância

var(df\_1$'Taxas de juros')

[1] 0.001983333

# Menor valor (mínino)

min(df\_1$'Taxas de juros')

[1] 2.5

# Maior valor (máximo)

max(df\_1$'Taxas de juros')

[1] 2.64

# Resumindo os dados

# Em colunas/vetores numéricos a função summary() fornece informações como menor valor (mínimo), máximo valor, mediana e a média, além dos valores que representam o primeiro e o terceiro quartis.

summary (df\_1$'Taxas de juros', digits = 3)

# Min. :2.50

# 1st Qu.:2.58

# Median :2.60

# Mean :2.60

# 3rd Qu.:2.63

# Max. :2.64

# “Plotando” os gráficos:

boxplot(df\_1$'Taxas de juros', xlab = "Número da Ação", ylab = "Taxa de Juros", main = "Taxas de Juros Recebidas em Ações", col = "orange")

# ou ...

boxplot(df\_1$'Taxas de juros', ylab = "Número da Ação", xlab = "Taxa de Juros", main = "Taxas de Juros Recebidas em Ações", col = "orange", horizontal = T)

plot(df\_1$'Taxas de juros', main = "Taxas de Juros Recebidas em Ações",

xlab = "Número da Ação",

ylab = "Taxa de Juros")

points(df\_1$'Taxas de juros', cex = .5, col = " red")

lines(df\_1$tx.juros, col = "dark red")

hist(df\_1$'Taxas de juros', main = "Taxas de Juros Recebidas em Ações",

ylab = "Número da Ação",

xlab = "Taxa de Juros")

barplot(df\_1$'Taxas de juros', main = "Taxas de Juros Recebidas em Ações",

xlab = "Número da Ação",

ylab = "Taxa de Juros")

# Obs:

# Pra utilizar nome de variáveis (colunas) com espaço, exige o uso de "" (aspas duplas) ou aspas simples ' ' no nome da variável (no exemplo, df\_1$'Taxas de juros').

# Para simplificar, pode-se “trocar” (modificar) o nome da variável para uma versão mais simples de se lembrar e utilizar.

# Exemplo:

names(df\_1) = "tx.juros"

View(df\_1)

df\_1

# A tibble: 10 × 1

# tx.juros

# <dbl>

# 1 2.59

# 2 2.64

# 3 2.6

# 4 2.62

# 5 2.57

# 6 2.55

# 7 2.61

# 8 2.5

# 9 2.63

#10 2.64

# “Plotando” os gráficos:

plot(df\_1$tx.juros, main = "Taxas de Juros Recebidas em Ações",

xlab = "Número da Ação",

ylab = "Taxa de Juros")

points(df\_1$tx.juros, cex = .5, col = " red")

lines(df\_1$tx.juros, col = "dark red")

hist(df\_1$tx.juros, main = "Taxas de Juros Recebidas em Ações",

ylab = "Número da Ação",

xlab = "Taxa de Juros")

barplot(df\_1$tx.juros, main = "Taxas de Juros Recebidas em Ações",

xlab = "Número da Ação",

ylab = "Taxa de Juros")

# Referências

## # Fonte: <<https://www.w3schools.com/r/r_math.asp>>