DSBD - Linguagens de Programação - R

Helena R. S. D'Espindula

2024-05-03

Contents

R Base **2**

Aritimética Básica com o R **8**

Operações Matemáticas 8

Operações Lógicas 10

Estruturas de Dados em R **12**

Tipos de Vetores 13

Sequências & Repetições 17

Seleção de Elementos 18

Modificar e Adcionar Elementos 20

Operações Estatísticas 21

Matrizes 23

Seleção de Elementos em Matrizes 24

Data Frames 26

Considerações Finais 27

Exercícios 27

Estruturas de Programação **29**

Estruturas de Programação 29

```
knitr::opts_chunk$set(echo = TRUE)

library(ggplot2)
#library(BiocManager)
```

R Base

Livros:

- [https://rstudio-education.github.io/hopr/]
- https://r4ds.had.co.nz/
- https://www.amazon.com/Hands-Data-Science-Techniques-manipulation-ebook/dp/B07FDD1KWJ
- https://r-graphics.org/
- https://adv-r.hadley.nz/
- https://ggplot2-book.org/
- https://www.tmwr.org/

Primeiro código em R

- Tudo após o # é um comentário

```
## Somando dois números
2 + 2
```

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

```
## Quantos segundos tem uma hora?
## Cada minuto tem 60 segundos,
## cada hora tem 60 minutos
60 * 60
```

```
## [1] 3600
```

Atribuição

- Para atribuírmos valores aos objetos, utilizamos o operador `<-` ou `=`
- Para inserir o operador `<-` no RStudio, pressione `Alt` + `-`

```
## Atribuindo um valor a um objeto
x <- 2
x
```

```
## [1] 2
```

```
y <- 3
y
```

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

Área de Trabalho

- Para visualizar os objetos criados, podemos utilizar a função `ls()`
- Para remover um objeto, utilizamos a função `rm()`

```
## Lista objetos
ls()
```

```
## [1] "x" "y"
```

```
## Remove objetos
rm(x)

## Lista objetos
ls()
```

```
## [1] "y"
```

```
## Apaga todos os objetos
rm(list = ls())

## Lista objetos
ls()
```

```
## character(0)
```

Ambientes na Área de Trabalho

- A função mostra o conteúdo da área de trabalho em `.GlobalEnv`
- Cada biblioteca (pacote) carregado cria um novo ambiente de trabalho (namespace)
- `search()` retorna a lista de espaços de trabalho

```
ls()
```

```
## character(0)
```

```
# Busca
search()
```

```
##  [1] ".GlobalEnv"        "package:ggplot2"   "package:stats"
##  [4] "package:graphics"  "package:grDevices" "package:utils"
##  [7] "package:datasets"  "package:methods"   "Autoloads"
## [10] "package:base"
```

```
# Lista o conteúdo de um pacote
ls("package:datasets")
```

```
##   [1] "ability.cov"           "airmiles"              "AirPassengers"
##   [4] "airquality"            "anscombe"              "attenu"
##   [7] "attitude"              "austres"               "beaver1"
##  [10] "beaver2"               "BJsales"               "BJsales.lead"
##  [13] "BOD"                   "cars"                  "ChickWeight"
##  [16] "chickwts"              "co2"                   "CO2"
##  [19] "crimtab"               "discoveries"           "DNase"
```

```
##  [22] "esoph"                "euro"                 "euro.cross"
##  [25] "eurodist"             "EuStockMarkets"       "faithful"
##  [28] "fdeaths"              "Formaldehyde"         "freeny"
##  [31] "freeny.x"             "freeny.y"             "HairEyeColor"
##  [34] "Harman23.cor"         "Harman74.cor"         "Indometh"
##  [37] "infert"               "InsectSprays"         "iris"
##  [40] "iris3"                "islands"              "JohnsonJohnson"
##  [43] "LakeHuron"            "ldeaths"              "lh"
##  [46] "LifeCycleSavings"     "Loblolly"             "longley"
##  [49] "lynx"                 "mdeaths"              "morley"
##  [52] "mtcars"               "nhtemp"               "Nile"
##  [55] "nottem"               "npk"                  "occupationalStatus"
##  [58] "Orange"               "OrchardSprays"        "PlantGrowth"
##  [61] "precip"               "presidents"           "pressure"
##  [64] "Puromycin"            "quakes"               "randu"
##  [67] "rivers"               "rock"                 "Seatbelts"
##  [70] "sleep"                "stack.loss"           "stack.x"
##  [73] "stackloss"            "state.abb"            "state.area"
##  [76] "state.center"         "state.division"       "state.name"
##  [79] "state.region"         "state.x77"            "sunspot.month"
##  [82] "sunspot.year"         "sunspots"             "swiss"
##  [85] "Theoph"               "Titanic"              "ToothGrowth"
##  [88] "treering"             "trees"                "UCBAdmissions"
##  [91] "UKDriverDeaths"       "UKgas"                "USAccDeaths"
##  [94] "USArrests"            "UScitiesD"            "USJudgeRatings"
##  [97] "USPersonalExpenditure" "uspop"               "VADeaths"
## [100] "volcano"              "warpbreaks"           "women"
## [103] "WorldPhones"          "WWWusage"
```

```
women #parte do pacote datasets
```

```
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

```
women <- 10986 #criei uma local

women # local tem prioridade
```

```
## [1] 10986
```

```r
datasets::women # especifica origem
```

```
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

```r
# Se for uma função "escondida" usa 3x dois pontos
# wTO:::CN_aux
```

Diretórios e Arquivos

- O diretório de trabalho é o local onde o R está apontando. Isto é, onde ele está lendo e salvando os arquivos - por padrão.
- Para saber qual é o diretório de trabalho, utilize a função `getwd()`.
- Para mudar o diretório de trabalho, utilize a função `setwd()`.
- Para listar o conteúdo de um diretório, utilize a função `list.files()` ou `dir()`

Arquivos da linguagem R

- Os arquivos da linguagem R possuem a extensão `.R`.
- `.RData` é um arquivo binário que contém todos os objetos da área de trabalho.
- `.Rhistory` é um arquivo que contém o histórico de comandos executados.
- `.Rprofile` é um arquivo que contém comandos que são executados toda vez que o R é iniciado.
- Pode ser utilizado para carregar pacotes, por exemplo.
- `.Renviron` é um arquivo que contém variáveis de ambiente.
- `.Rproj` é um arquivo que contém as configurações do projeto.

Pacotes

- Pacotes são coleções de funções, dados e documentação que ampliam as funcionalidades do R.
- Para instalar um pacote da CRAN, utilize a função `install.packages()`
- Para carregar um pacote, utilize a função `library()`
- Pacotes podem estar disponíveis em outros repositórios, como o Bioconductor e GitHub.

```r
# Instalar o pacote
#install.packages("ggplot2")
```

```r
# Carregar o pacote
library(ggplot2)

# Verificar o conteúdo
head(ls("package:ggplot2"), 30)

# Documentação do pacote
#help(package = "ggplot2")

# Caminhos de instalação
.libPaths()

# Remover o pacote da sessão
#("package:ggplot2", unload = TRUE)
```

Instalando do Biocondutor

- https://www.bioconductor.org/

```r
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install(version = "3.19")

BiocManager::install("msdata")
# Various Mass Spectrometry raw data example files
```

Buscando Ajuda

- Para acessar a documentação de uma função, utilize a função `help()` ou `?`
- `??`, `help.search` e `apropos()` são funções que buscam por termos na documentação.
- Para acessar a documentação de um pacote, utilize a função 'help(package = "nome_do_pacote"
- Muitos pacotes possuem tutoriais e exemplos de uso em suas vinhetas. Para acessar, utilize a função `vignette()`

```r
# Buscando pela documentação
#?mean
#help(mean)

# Buscando por termos
#??mean
#help.search("mean")
#apropos("mean")

# Vignettes
#browseVignettes(package = "ggplot2")

# Busca pelo termo no r-project.org
#RSiteSearch("mean")
```

Campos da Documentação

- Title: Título da função.

- Description: Descrição da função.
- Usage: Como utilizar a função.
- Arguments: Argumentos da função.
- Details: Detalhes da função.
- Value: O que a função retorna.
- See Also: Funções relacionadas.
- Examples: Exemplos de uso.
- Author: Autor da função.
- References: Referências bibliográficas.

Manipulação de expressões e diferenciação de letras maiúsculas e minúsculas

- O R é case-sensitive, isto é, ele diferencia letras maiúsculas de minúsculas.
- Nomes de Variáveis e Funções: Ao nomear suas variáveis e funções, separe palavras com underscore (_) ou camelCase. Por exemplo: nome_variavel, meu_contador, calcularMedia, etc.
- Palavras Reservadas: O R possui palavras reservadas que são usadas para funções ou operações internas e não devem ser usadas como nomes de objetos.
  - Exemplo: `if`, `else`, `while`, `function`, `for`, `NA`, `TRUE`, entre outras.
- Boas Práticas: Evite usar palavras reservadas e nomes de funções nativas como nomes de variáveis.

Exemplos

```
# Exemplo de manipulação de expressões
x <- 5
y<-10 # Não recomendado, espaços são mais legíveis
z <- (x +
y) # Continuando o comando em outra linha
x # imprime na tela o valor de x
```

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

```
y # imprime o valor de y
```

```
## [1] 10
```

```
z # imprime o valor de z
```

```
## [1] 15
```

```
# Exemplo de diferenciação de letras maiúsculas e minúsculas
variavel <- 42
Variavel <- 24

variavel
```

```
## [1] 42
```

```
Variavel
```

```
## [1] 24
```

Aritimética Básica com o R

Operações Matemáticas

Operações Básicas

```
1 + 1 # Adição
```

```
## [1] 2
```

```
2 - 1 # Subtração
```

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

```
3 * 2 # Multiplicação
```

```
## [1] 6
```

```
4 / 2 # Divisão
```

```
## [1] 2
```

```
5 ^ 2 # Potenciação
```

```
## [1] 25
```

```
5 %% 2 # Resto da divisão
```

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

```
5 %/% 2 # Divisão inteira
```

```
## [1] 2
```

Logarítimo

```
log(10) # Logarítimo natural
```

```
## [1] 2.302585
```

```r
exp(10) # Exponencial
```

```
## [1] 22026.47
```

```r
log10(100) # Logarítimo base 10
```

```
## [1] 2
```

```r
log2(10) # Logarítimo base 2
```

```
## [1] 3.321928
```

```r
log(100, base = 8) # Log. base 8
```

```
## [1] 2.214619
```

Funções Trigonométricas

```r
sin(0) # Seno
```

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

```r
cos(0) # Cosseno
```

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

```r
tan(0) # Tangente
```

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

```r
asin(0) # Arco seno
```

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

```r
acos(0) # Arco cosseno
```

```
## [1] 1.570796
```

```r
atan(0) # Arco tangente
```

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

Arrerondamento

```r
round(pi, digits = 2) # Arredondament
```

```
## [1] 3.14
```

```r
ceiling(pi) # Teto
```

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

```r
floor(pi) # Piso
```

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

```r
trunc(pi) # Truncamento
```

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

Operações Lógicas

Comparativos

```r
1 == 1 # Igual
```

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

```r
1 != 2 # Diferente
```

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

```r
1 > 2 # Maior
```

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

```r
1 < 2 # Menor
```

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

```r
1 >= 1 # Maior ou Igual
```

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

```r
2 >= 1 # Menor ou Igual
```

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

```r
a <- 1
b <- 2
a <= b
```

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

Operadores Lógicos

```r
(1 == 1) & (2 == 2) # E / AND
```

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

```r
(1 == 1) | (2 == 3) # OU / OR
```

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

```r
!(1 == 2) # NÃO / NOT
```

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

Strings

```r
"R" == "r" # Comparação de strings
```

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

```r
"a" < "b" # Ordem Alfanumérica
```

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

```r
"1" < "2" # Ordem Alfanumérica
```

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

Tipos Especiais

- `NA`: Valores Ausentes
- `NaN`: Not a Number
- `Inf` e `-Inf`: Infinito
- `NULL`: Valor Nulo (vazio)

Estruturas de Dados em R

Vetores

- Um vetor, em R, é uma sequência de elementos do mesmo tipo.
- Para criar um vetor, utilize a função `c()`

```
numeros <- c(1, 2, 3, 4, 5)
numeros
```

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

```
letras <- c("a", "b", "c", "d", "e")
letras
```

```
## [1] "a" "b" "c" "d" "e"
```

```
logicos <- c(TRUE, FALSE, TRUE, FALSE)
logicos
```

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

```
## Coerção
vetor <- c(numeros, letras, logicos)
vetor
```

```
##  [1] "1"     "2"     "3"     "4"     "5"     "a"     "b"     "c"     "d"
## [10] "e"     "TRUE"  "FALSE" "TRUE"  "FALSE"
```

- Vetores em R começam com o índice 1.
- Para acessar um elemento de um vetor, utilize `[]`.

```
# Acessa o primeiro elemento
numeros[1]
```

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

```
# Acessa o terceiro elemento
letras[3]
```

```
## [1] "c"
```

```
# Acessa o segundo elemento
logicos[2]
```

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

```r
# Acessa o quinto elemento
vetor[5]
```

```
## [1] "5"
```

Tipos de Vetores

- Numérico: `numeric`
  - `c(1.2, 2.5, 3.14)`
- Inteiro: `integer`
  - `c(1, 2, 3, 4, 5)`
- Lógico: `logical`
  - `c(TRUE, FALSE, FALSE)`
- Complexo: `complex`
  - `c(1 + 2i, 2 + 0i)`
- Caracter: `character`
  - `c("a", "b", "c")`
- Factor: `factor`
  - `factor(c("Tipo 1", "Tipo 2", "Tipo 3"))`

Atenção: Se, misturar tipos vai virar char.

Consultando Tipos

```r
# Funções que começam com `is.`
# retornam TRUE ou FALSE
# Lista funções is.
apropos("^is\\.")
```

```
##  [1] "is.array"               "is.atomic"
##  [3] "is.call"                "is.character"
##  [5] "is.complex"             "is.Coord"
##  [7] "is.data.frame"          "is.double"
##  [9] "is.element"             "is.empty.model"
## [11] "is.environment"         "is.expression"
## [13] "is.facet"               "is.factor"
## [15] "is.finite"              "is.finite.POSIXlt"
## [17] "is.function"            "is.ggplot"
## [19] "is.ggproto"             "is.hashtab"
## [21] "is.infinite"            "is.infinite.POSIXlt"
## [23] "is.integer"             "is.language"
## [25] "is.leaf"                "is.list"
## [27] "is.loaded"              "is.logical"
## [29] "is.matrix"              "is.mts"
## [31] "is.na"                  "is.na.data.frame"
```

```
## [33] "is.na.numeric_version"   "is.na.POSIXlt"
## [35] "is.na<-"                 "is.na<-.default"
## [37] "is.na<-.factor"          "is.na<-.numeric_version"
## [39] "is.name"                 "is.nan"
## [41] "is.nan.POSIXlt"          "is.null"
## [43] "is.numeric"              "is.numeric.Date"
## [45] "is.numeric.difftime"     "is.numeric.POSIXt"
## [47] "is.numeric_version"      "is.object"
## [49] "is.ordered"              "is.package_version"
## [51] "is.pairlist"             "is.primitive"
## [53] "is.qr"                   "is.R"
## [55] "is.raster"               "is.raw"
## [57] "is.recursive"            "is.relistable"
## [59] "is.single"               "is.stepfun"
## [61] "is.symbol"               "is.table"
## [63] "is.theme"                "is.ts"
## [65] "is.tskernel"             "is.unsorted"
## [67] "is.vector"
```

```
# Verifica se é inteiro
is.integer(numeros)
```

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

```
# Verifica se é numérico
is.numeric(numeros)
```

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

```
# Verifica se é caracter
is.character(letras)
```

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

```
# Verifica se é lógico
is.logical(logicos)
```

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

```
fator = factor( c("Tipo 1", "Tipo 2", "Tipo 3"))

# Verifica se é fator
is.factor(fator)
```

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

Conversão de Tipos

```r
# Funções que começam com `as.`
# Lista funções as.
head(apropos("^as\\."),15)
```

```
##  [1] "as.array"                     "as.array.default"
##  [3] "as.call"                      "as.character"
##  [5] "as.character.condition"       "as.character.Date"
##  [7] "as.character.default"         "as.character.error"
##  [9] "as.character.factor"          "as.character.hexmode"
## [11] "as.character.numeric_version" "as.character.octmode"
## [13] "as.character.POSIXt"          "as.character.srcref"
## [15] "as.complex"
```

```r
# Converte para caracter
as.character(numeros)
```

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

```r
# Converte para numérico
as.numeric(fator)
```

```
## [1] 1 2 3
```

```r
datas <- c("2021-01-01", "2021-01-02", "2021-01-03")
# Converte para data
as.Date(datas)
```

```
## [1] "2021-01-01" "2021-01-02" "2021-01-03"
```

```r
datas
```

```
## [1] "2021-01-01" "2021-01-02" "2021-01-03"
```

Verificando Tipos

- Quando não sabemos a classe de um objeto, podemos utilizar a função `class()`.

```r
#class()
class(numeros)
```

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

```r
class(letras)
```

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

```r
class(logicos)
```

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

```r
class(datas)
```

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

Métodos

- Métodos são funções genéricas que atuam conforme a classe do objeto.
- Para saber quais métodos estão disponíveis para uma classe, utilize a função `methods(class = "classe")`

```r
methods(class = "numeric")
```

```
##  [1] all.equal     as.data.frame as.Date       as.POSIXct    as.POSIXlt
##  [6] as.raster     coerce        glyphJust     Ops           scale_type
## see '?methods' for accessing help and source code
```

```r
methods(class = "character")
```

```
##  [1] all.equal                as.data.frame            as.Date
##  [4] as.POSIXlt               as.raster                coerce
##  [7] coerce<-                 formula                  getDLLRegisteredRoutines
## [10] glyphJust                Ops                      scale_type
## see '?methods' for accessing help and source code
```

```r
## Podemos ter um vetor numérico com
altura <- c("João" = 1.82,
            "Bianca" = 1.68,
            "Eduarda" = 1.62)
class(altura)
```

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

```r
attributes(altura)
```

```
## $names
## [1] "João"    "Bianca"  "Eduarda"
```

```r
names(altura)
```

```
## [1] "João"    "Bianca"  "Eduarda"
```

Sequências & Repetições

- Gerar sequências:
  - `seq(from = x, to = y, by = z)`
- Repetir elementos:
  - `rep(x, times = n)`
  - `rep(x, each = n)`

```
# Sequência de 1 a 10
seq1 <- c(1, 10)
seq1
```

```
## [1]  1 10
```

```
seq
```

```
## function (...)
## UseMethod("seq")
## <bytecode: 0x00000268c253bf40>
## <environment: namespace:base>
```

```
# [1] 1 2 3 4 5 6 7 8 9 10

# Sequência de 10 a 1, de 2 em 2
seq2 <- c(10, 1, by = -2)
seq2
```

```
##       by
## 10  1 -2
```

```
seq
```

```
## function (...)
## UseMethod("seq")
## <bytecode: 0x00000268c253bf40>
## <environment: namespace:base>
```

```
# [1] 10 8 6 4 2

# Repete 1, 2 e 3, 3 vezes
rep1 <- rep( c(1, 2, 3), times = 3)
rep1
```

```
## [1] 1 2 3 1 2 3 1 2 3
```

```
#[1] 1 2 3 1 2 3 1 2 3

# Repete 1, 2 e 3, 3 vezes cada
rep2 <- rep(c(1, 2, 3), each = 3)
rep2
```

```
## [1] 1 1 1 2 2 2 3 3 3
```

```
#[1] 1 1 1 2 2 2 3 3 3
```

Números Aleatórios

- Números aleatórios:
  - runif()
  - rnorm()
  - sample()

```
set.seed(123) #só para ser sempre os mesmos

# 5 Números aleatórios entre 0 e 1
runif(5)
```

```
## [1] 0.2875775 0.7883051 0.4089769 0.8830174 0.9404673
```

```
# [1] 0.72837130 0.86886759 0.04289164 0.47154488 0.70211836

# 5 Números aleatórios de uma normal, com média 0 e desvio padrão 1
rnorm(5)
```

```
## [1] -1.6895557  1.2394959 -0.1089660 -0.1172420  0.1830826
```

```
# [1] -0.47879958 0.06497995 -0.48012593 1.19685144 0.69212656

# Amostras aleatórias
sample(numeros,
size = 3,
replace = FALSE)
```

```
## [1] 3 4 1
```

```
# [1] 5 2 3

sample(letras,
size = 5,
replace = TRUE)
```

```
## [1] "c" "c" "a" "d" "a"
```

```
# [1] "d" "a" "b" "e" "a"
```

Seleção de Elementos

Seleção Posicional

```r
alturas <- c("João" = 1.82,
             "Bianca" = 1.68,
             "Carlos" = 1.75,
             "Ana" = 1.70)

# Seleciona o primeiro elemento
print("Seleciona o primeiro elemento")
```

```
## [1] "Seleciona o primeiro elemento"
```

```r
alturas[1]
```

```
## João
## 1.82
```

```r
# Seleciona até o terceiro elemento
print("Seleciona até o terceiro elemento")
```

```
## [1] "Seleciona até o terceiro elemento"
```

```r
alturas[1:3]
```

```
##   João Bianca Carlos
##   1.82   1.68   1.75
```

```r
# Seleciona elementos 1, 3 e 4
print("Seleciona elementos 1, 3 e 4")
```

```
## [1] "Seleciona elementos 1, 3 e 4"
```

```r
alturas[c(1, 3, 4)]
```

```
##   João Carlos    Ana
##   1.82   1.75   1.70
```

```r
# remove o segundo elemento
print("remove o segundo elemento")
```

```
## [1] "remove o segundo elemento"
```

```r
alturas[-2]
```

```
##   João Carlos    Ana
##   1.82   1.75   1.70
```

Seleção Condicional

```r
# Seleciona alturas maiores que 1.70
mascara_logica <- alturas > 1.70
alturas[mascara_logica]
```

```
##   João Carlos
##   1.82   1.75
```

```r
alturas[alturas > 1.70]
```

```
##   João Carlos
##   1.82   1.75
```

Seleção por Nome

```r
# Seleciona a altura de João
alturas["João"]
```

```
## João
## 1.82
```

```r
# Seleciona a altura de João e Ana
alturas[c("João", "Ana")]
```

```
## João  Ana
## 1.82 1.70
```

Modificar e Adcionar Elementos

Modificar

- Podemos selecionar o elemento de interesse e então atribuir um novo valor

```r
alturas
```

```
##   João Bianca Carlos    Ana
##   1.82   1.68   1.75   1.70
```

```r
# Modifica a altura de João
alturas["João"] <- 1.85
alturas
```

```
##   João Bianca Carlos    Ana
##   1.85   1.68   1.75   1.70
```

```r
# Atribui altura desconhecida a Bianc
alturas["Bianca"] <- NA
alturas
```

```
##   João Bianca Carlos    Ana
##   1.85     NA   1.75   1.70
```

```
# Remove a altura de Carlos
alturas = alturas[-3]
```

Adicionar

```
# Adiciona a altura de Ivete
append(alturas, value = c("Ivete", 1.60))
```

```
##    João  Bianca     Ana
##  "1.85"      NA   "1.7" "Ivete"   "1.6"
```

```
# Adiciona a altura de Anderson no in
append(alturas, value = c("Anderson", 1.75), after = 0)
```

```
##                             João     Bianca        Ana
## "Anderson"     "1.75"     "1.85"          NA      "1.7"
```

```
# Concatena alturas
alturas2 <- c("Alana" = 1.70, "Rafael" = 1.80)
alturas <- c(alturas, alturas2)
alturas
```

```
##   João Bianca    Ana  Alana Rafael
##   1.85     NA   1.70   1.70   1.80
```

Operações Estatísticas

```
y <- c(7, 5, 2, 2, 4, 8, 5, 2, 6, 4, 5, 10, 3, 2, 6, 10, 7, 8, 6, 10, 3, 4, 5, 1)
y
```

```
##  [1]  7  5  2  2  4  8  5  2  6  4  5 10  3  2  6 10  7  8  6 10  3  4  5  1
```

```
# Número de elementos
length(y)
```

```
## [1] 24
```

```
#[1] 24

# Soma dos elementos
sum(y)
```

```
## [1] 125
```

```
#[1] 125

# Média
mean(y)
```

```
## [1] 5.208333
```

```
#[1] 5.208333

# Mediana
median(y)
```

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

```
#[1] 5

# Máximo
max(y)
```

```
## [1] 10
```

```
#[1] 10

# Mínimo
min(y)
```

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

```
#[1] 1

# Variância
var(y)
```

```
## [1] 7.21558
```

```
#[1] 7.21558

# Desvio padrão
sd(y)
```

```
## [1] 2.686183
```

```
#[1] 2.686183

# Desvio absoluto mediana
mad(y, constant = 1)
```

```
## [1] 2
```

```r
#[1] 2

# Coeficiente de variação
100 * sd(y) / mean(y)
```

```
## [1] 51.57472
```

```r
#[1] 51.57472

# Quartis
quantile(y)
```

```
##   0%  25%  50%  75% 100%
##    1    3    5    7   10
```

```r
#0 % 25% 50 % 75% 100%
#1 3 5 7 10

# Amplitude Interquartilica
IQR(y)
```

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

```r
#[1] 4

# Tabela de Frequência
table(y)
```

```
## y
##  1  2  3  4  5  6  7  8 10
##  1  4  2  3  4  3  2  2  3
```

```r
prop.table(table(y))
```

```
## y
##          1          2          3          4          5          6          7
## 0.04166667 0.16666667 0.08333333 0.12500000 0.16666667 0.12500000 0.08333333
##          8         10
## 0.08333333 0.12500000
```

Matrizes

- São bidimensionais, com linhas e colunas.
- Todos elementos são do mesmo tipo.

```r
# Cria uma matriz 2x3
matriz <- matrix(1:6, nrow = 2)
matriz
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
# Cria uma matriz 3x2
matriz2 <- matrix(1:6, ncol = 2)
matriz2
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
# Criando matriz a partir de um vetor
vet1 <- c(1, 2, 3)
vet2 <- c(4, 5, 6)

# Junta vet1 e vet2 por linha (row)
matriz3 <- rbind(vet1, vet2)
matriz3
```

```
##      [,1] [,2] [,3]
## vet1    1    2    3
## vet2    4    5    6
```

```
# Junta vet1 e vet2 por coluna (collum)
matriz4 <- cbind(vet1, vet2)
matriz4
```

```
##      vet1 vet2
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

Seleção de Elementos em Matrizes

Seleção de Elementos em Matrizes

```
# Seleciona o elemento da primeira li
# e segunda coluna
matriz[1, 2]
```

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

```
#[1] 3

# Seleciona a primeira linha
matriz[1, ]
```

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

```r
#[1] 1 3 5

# Seleciona a segunda coluna
matriz[, 2]
```

```
## [1] 3 4
```

```r
#[1] 3 4
```

Operação com Matrizes

```r
mat1 <- matrix(1:4, nrow = 2)
mat2 <- matrix(5:8, nrow = 2)

# Soma de matrizes
mat1 + mat2
```

```
##      [,1] [,2]
## [1,]    6   10
## [2,]    8   12
```

```r
matrix
```

```
## function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
## {
##     if (is.object(data) || !is.atomic(data))
##         data <- as.vector(data)
##     .Internal(matrix(data, nrow, ncol, byrow, dimnames, missing(nrow),
##         missing(ncol)))
## }
## <bytecode: 0x00000268bf0605d8>
## <environment: namespace:base>
```

```r
# Multiplicação de matrizes
mat1 %*% mat2
```

```
##      [,1] [,2]
## [1,]   23   31
## [2,]   34   46
```

```r
# Multiplicação por escalar
mat1 * 2
```

```
##      [,1] [,2]
## [1,]    2    6
## [2,]    4    8
```

```r
# Transposta
t(mat1)
```

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

```r
colSums(mat1)
```

```
## [1] 3 7
```

```r
rowSums(mat1)
```

```
## [1] 4 6
```

```r
colMeans(mat1)
```

```
## [1] 1.5 3.5
```

```r
rowMeans(mat1)
```

```
## [1] 2 3
```

Data Frames

- Estrutura de dados mais comum em R
- Semelhante a uma planilha
- Cada coluna pode ter um tipo diferente
- Cada linha é uma observação
- Podemos criar um data frame a partir de vetores.

```r
# Cria um data frame
nomes = c("Joao", "Ana", "Carlos")
idades = c(25, 30, 22)
alturas = c(1.75, 1.60, 1.80)

df <- data.frame(nome = nomes,
                 idade = idades,
                 altura = alturas)
df
```

```
##     nome idade altura
## 1   Joao    25   1.75
## 2    Ana    30   1.60
## 3 Carlos    22   1.80
```

```r
#df2 <- (nome = c("Joao", "Ana", "Carlos"), idade = c(25, 30, 22), altura = c(1.75, 1.60, 1.80))
#df2

summary(df)
```

```
##      nome               idade           altura
##  Length:3           Min.   :22.00   Min.   :1.600
##  Class :character   1st Qu.:23.50   1st Qu.:1.675
##  Mode  :character   Median :25.00   Median :1.750
##                     Mean   :25.67   Mean   :1.717
##                     3rd Qu.:27.50   3rd Qu.:1.775
##                     Max.   :30.00   Max.   :1.800
```

```
names(df)
```

```
## [1] "nome"   "idade"  "altura"
```

```
#[1] "nome" "idade" "altura"

df$nome
```

```
## [1] "Joao"   "Ana"    "Carlos"
```

```
#[1] "João" "Ana" "Carlos"
```

Considerações Finais

- Vetores, matrizes e data frames são estruturas de dados fundamentais em R.
- Vetores são unidimensionais, matrizes são bidimensionais e data frames são semelhantes a planilhas.
- Podemos realizar operações estatísticas e matemáticas com essas estruturas.

Exercícios

1. Crie um vetor com 10 números inteiros aleatórios entre 1 e 100.
2. Calcule a média, mediana, variância e desvio padrão do vetor criado.

```
set.seed(123)

vetor <- sample.int(100, 10)
vetor
```

```
##  [1] 31 79 51 14 67 42 50 43 97 25
```

```
mean(vetor)
```

```
## [1] 49.9
```

```
median(vetor)
```

```
## [1] 46.5
```

```r
sd(vetor)
```

```
## [1] 25.24304
```

3. Crie uma matriz 3x3 com números inteiros de 1 a 9.
4. Selecione o elemento da segunda linha e terceira coluna da matriz criada.

```r
matrix2 <- matrix(1:9, nrow = 3, ncol = 3, byrow = FALSE)
# faz por coluna, é o defaut
matrix2
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```r
matrix1 <- matrix(1:9, nrow = 3, ncol = 3, byrow = TRUE)
# para fazer por linha
matrix1
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

```r
matrix1[2, 3]
```

```
## [1] 6
```

5. Crie um data frame com 5 observações e 3 variáveis: nome, idade e altura.

```r
nomes = c("Joao", "Ana", "Carlos", "Andre", "Bia")
idades = c(25, 30, 22, 45, 33)
alturas = c(1.75, 1.60, 1.80, 1.56, 1.97)

df <- data.frame(nome = nomes,
                 idade = idades,
                 altura = alturas)
df
```

```
##     nome idade altura
## 1   Joao    25   1.75
## 2    Ana    30   1.60
## 3 Carlos    22   1.80
## 4  Andre    45   1.56
## 5    Bia    33   1.97
```

```r
mean(df$idade)
```

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

Estruturas de Programação

Estruturas de Programação

Existem duas estruturas princiais de programação: controle e repetição.

Estruturas de Controle

- Permitem a execução de um bloco de código baseado em uma condição testada.
- If - Else: Executa um bloco de código se a condição for verdadeira.
- Switch: Executa um bloco de código baseado em múltiplas condições.

Estruturas de Repetição

- Permitem a execução de um bloco de código repetidamente.
- For: Executa um bloco de código um número específico de vezes.
- While: Executa um bloco de código enquanto uma condição for verdadeira.
- Repeat: Executa um bloco de código indefinidamente.

If-Else

- A estrutura if - else permite a execução de um bloco de código baseado em uma condição.
- Se a condição for verdadeira, o bloco de código dentro do if é executado.
- Se a condição for falsa, o bloco de código dentro do else é executado.

```
if(condicao) {
  # Bloco de código
} else {
  # Bloco de código
}
```

```
nota <- 7
if(nota >= 7) {
  print("Aprovado")
} else {
  print("Reprovado")
}
```

```
## [1] "Aprovado"
```

Exemplo de If - Else

- Implemente um código que envia um e-mail automaticamente para os clientes de uma empresa que fizeram uma compra de acordo com dois critérios: Horário da compra e valor da compra.
  - Se a compra for pela manhã, o e-mail deve ser iniciar com Bom dia.
  - Se a compra for à tarde, o e-mail deve ser iniciar com Boa tarde.
  - Se a compra for à noite, o e-mail deve ser iniciar com Boa noite.
  - Se o valor da compra for maior que R$100, o e-mail deve conter um cupom de desconto de 10%.
  - Se o valor da compra for menor ou igual a R$100, o e-mail deve conter um cupom de desconto de 5%.

```r
hora_compra <- "manha"
valor_compra <- 150
if(hora_compra == "manha" & valor_compra > 100) {
  print("Bom dia! Você ganhou um cupom de 10% de desconto.")
} else if(hora_compra == "manha" & valor_compra <= 100) {
  print("Bom dia! Você ganhou um cupom de 5% de desconto.")
} else if(hora_compra == "tarde" & valor_compra > 100) {
  print("Boa tarde! Você ganhou um cupom de 10% de desconto.")
} else if(hora_compra == "tarde" & valor_compra <= 100) {
  print("Boa tarde! Você ganhou um cupom de 5% de desconto.")
} else if(hora_compra == "noite" & valor_compra > 100) {
  print("Boa noite! Você ganhou um cupom de 10% de desconto.")
} else {
  print("Boa noite! Você ganhou um cupom de 5% de desconto.")
}
```

```
## [1] "Bom dia! Você ganhou um cupom de 10% de desconto."
```