

PROGRAMAÇÃO EM R

Objetos em R

Explore os fundamentos dos objetos em R e aprenda como variáveis armazenam diferentes tipos de dados para análise e programação.



Eduardo Ogasawara

eduardo.ogasawara@cefet-rj.br

<https://eic.cefet-rj.br/~eogasawara>

OBJETOS E VALORES

O que é uma Variável?



Nome

Uma variável é um **nome associado a um valor**. Funciona como um rótulo que identifica um dado na memória.



Armazenamento

Esse valor fica armazenado na memória do computador, pronto para ser acessado quando necessário.



Reutilização

O nome permite acessar o valor, reutilizá-lo em diferentes partes do código e até modificá-lo quando necessário.

Pense em uma variável como uma caixa etiquetada: o nome é a etiqueta, e o valor é o que está dentro da caixa.

Objetos, Tipos e Valores

Em R, **tudo é objeto**. Esta é a filosofia central da linguagem.

1

Valor

Todo objeto contém um valor - pode ser um número, texto, ou estrutura mais complexa

2

Tipo

Cada objeto possui um tipo (numérico, caractere, lógico, etc.) que determina quais operações podem ser realizadas

3

Operações

Funções operam sobre objetos e produzem novos objetos como resultado

O que são Objetos?

Em R, variáveis são contêineres que armazenam objetos. Esses objetos podem assumir várias formas e estruturas, cada uma com seu propósito específico.



Vetores

Sequências de elementos do mesmo tipo



Atributos

Metadados associados aos objetos



Matrizes

Estruturas bidimensionais de dados



Classes

Tipos e categorias de objetos



Conversão

Transformação entre tipos



Listas

Coleções flexíveis de objetos

Vetores: A Base de Tudo

Criando um Vetor

A função `c()` combina elementos em um vetor. Vetores são a estrutura mais básica em R.

```
dado <- c(1, 2, 3, 4, 5, 6)  
dado  
## [1] 1 2 3 4 5 6  
  
is.vector(dado)  
## [1] TRUE
```



Este exemplo cria um vetor chamado `dado` que contém os números de 1 a 6, como as faces de um dado.

Vetores Unitários

Em R, mesmo valores únicos são tecnicamente vetores com comprimento 1. Esta característica torna a linguagem consistente e poderosa.

1

Criação

```
numero <- 5  
numero  
## [1] 5
```

2

Verificação

```
is.vector(numero)  
## [1] TRUE
```

3

Comprimento

```
length(numero)  
## [1] 1  
  
length(dado)  
## [1] 6
```

- Dica:** Use a função `length()` para descobrir quantos elementos existem em um vetor.

Tipos de Dados: Integer e Character

Integer (Inteiro)

Números inteiros são identificados pelo sufixo L. São mais eficientes em termos de memória.

```
inteiro <- 1L  
typeof(inteiro)  
## [1] "integer"
```

Character (Texto)

Strings ou textos são definidos entre aspas. Perfeitos para armazenar palavras e frases.

```
texto <- "ás"  
typeof(texto)  
## [1] "character"
```

Vetores e Funções

Vetores podem armazenar sequências de inteiros ou caracteres, e R oferece funções poderosas para manipulá-los.



Vetor Inteiro

```
cartas <- 1L:13L
```

Cria uma sequência de 1 a 13

Vetor de Caracteres

```
faces <- c("ás", "dois", "três",  
"quatro", "cinco", "seis", "sete",  
"oito", "nove", "dez", "valete",  
"dama", "rei")
```

Funções Aplicadas

```
n <- sum(cartas)  
is.integer(n)  
## [1] TRUE
```

```
m <- max(faces)  
m  
## [1] "valete"
```

Tipo Double (Decimal)

Por padrão, números em R são armazenados como **double** (ponto flutuante de dupla precisão), mesmo quando parecem inteiros.

```
dado <- c(1, 2, 3, 4, 5, 6)
 dado
## [1] 1 2 3 4 5 6

typeof(dado)
## [1] "double"
```

- O tipo double permite maior precisão em cálculos matemáticos e é o padrão para operações numéricas.

Tipo Logical (Lógico)

Valores lógicos são fundamentais para controle de fluxo e filtragem de dados. Resultam de comparações e operações booleanas.

>

Operadores de Comparação

- > maior que
- < menor que
- >= maior ou igual
- <= menor ou igual
- == igual
- != diferente

||
|

Exemplo Prático

```
3 > 4  
## [1] FALSE  
  
logico <- c(TRUE, FALSE, 3 >= 4,  
         3 < 4, 3 <= 4, 3 < 4,  
         3 != 4, 4 == 4)  
  
logico  
## [1] TRUE FALSE FALSE TRUE  
## TRUE TRUE TRUE TRUE  
  
typeof(logico)  
## [1] "logical"
```

Números Complexos

R suporta nativamente números complexos, essenciais para cálculos matemáticos e de engenharia avançados.

A parte imaginária é representada por `i`.

```
comp <- c(1 + 1i, 1 + 2i, 1 + 3i)
comp
## [1] 1+1i 1+2i 1+3i
```

```
typeof(comp)
## [1] "complex"
```

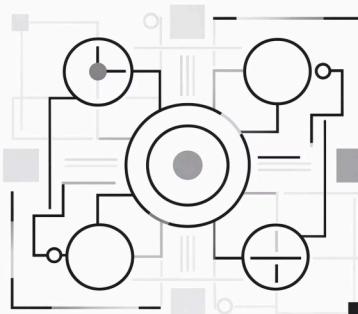


Números complexos são úteis em análise de sinais, processamento de imagens e computação científica.

Tipo Raw (Byte)

O tipo raw armazena dados brutos em bytes, útil para manipulação de dados binários de baixo nível.

```
119110101011010101101011001110  
111010101101011101001101010110  
111010101001101010101011010151  
119110101101011001101111001110  
111010101111001010101011010151
```



```
011010111011011010011011001110  
111001101000111010100111001110  
010110101011011001101101011101  
111001101111001011001010010110
```

Criando dados Raw

```
r <- raw(3)  
typeof(r)  
## [1] "raw"
```

Cria um vetor de 3 bytes inicializados com zero.

Manipulação

```
r[2] <- as.raw(255)  
r[3] <- as.raw(1024)  
## Warning: out-of-range  
## values treated as 0
```

```
r  
## [1] 00 ff 00
```

Valores são limitados a 0-255 (1 byte).

Atributos: Metadados dos Objetos

Atributos são informações adicionais anexadas aos objetos. O atributo `names` é um dos mais comuns.

1

Sem Atributos

```
dado <- c(1,2,3,4,5,6)
attributes(dado)
## NULL
```

2

Adicionando Names

```
names(dado) <- c("um", "dois",
"três", "quatro", "cinco", "seis")
attributes(dado)
## $names
## [1] "um" "dois" "três"
##   "quatro" "cinco" "seis"
```

3

Removendo Atributos

```
names(dado) <- NULL
dados
## [1] 1 2 3 4 5 6
```

Transformando em Matriz

Vetores podem ser transformados em matrizes alterando o atributo de dimensão. Isso reorganiza os dados em linhas e colunas.

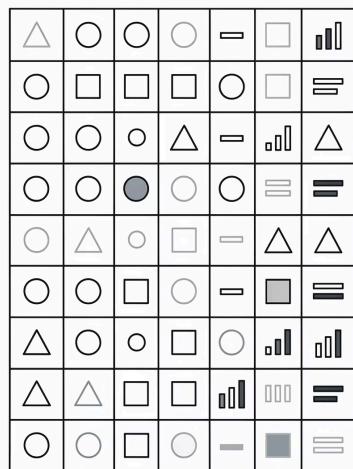
```
dado <- 1:6  
dim(dado) <- c(2, 3)  
 dado  
## [,1] [,2] [,3]  
## [1,] 1 3 5  
## [2,] 2 4 6
```

```
attributes(dado)  
## $dim  
## [1] 2 3
```

O vetor agora tem 2 linhas e 3 colunas. Os dados são preenchidos por coluna (verticalmente).

Criando Matrizes

A função `matrix()` oferece controle completo sobre como os dados são organizados em uma estrutura bidimensional.



Por Coluna (Padrão)

```
m <- matrix(dado, nrow = 2)  
m  
## [1] [,2] [,3]  
## [1,] 1 3 5  
## [2,] 2 4 6
```

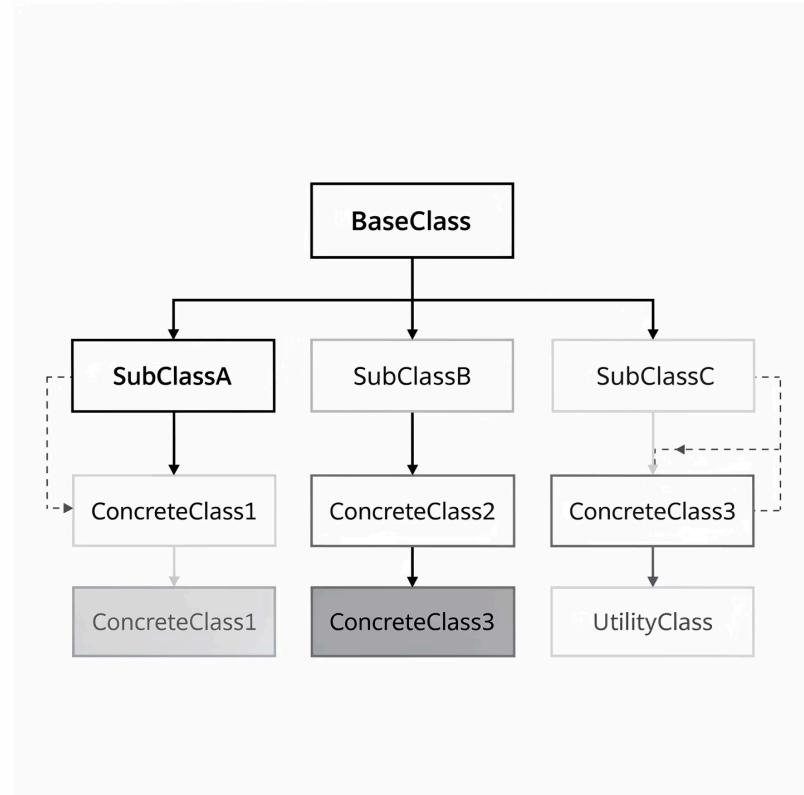
Preenche a matriz coluna por coluna

Por Linha

```
n <- matrix(dado, nrow = 2, byrow = TRUE)  
n  
## [1] [,2] [,3]  
## [1,] 1 2 3  
## [2,] 4 5 6
```

Usa `byrow = TRUE` para preencher linha por linha

Classes de Objetos



Todo objeto em R tem um **tipo** (como são armazenados) e uma **classe** (como se comportam).

```
typeof(dado)  
## [1] "integer"
```

```
class(dado)  
## [1] "matrix" "array"
```

```
attributes(dado)  
## $dim  
## [1] 2 3
```

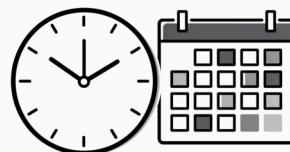
O tipo é **integer**, mas a classe é **matrix** devido ao atributo de dimensão.

Data e Hora (Timestamp)

R possui classes especializadas para trabalhar com datas e horários, essenciais para análise de séries temporais.

Capturando o Momento

```
now <- Sys.time()  
now  
## [1] "2026-01-08 23:42:12 -03"
```



Estrutura Interna

```
typeof(now)  
## [1] "double"  
  
class(now)  
## [1] "POSIXct" "POSIXt"
```

- ☐ Internamente é um número (segundos desde 1970), mas a classe `POSIXct` permite formatação e cálculos de tempo.

Dados Categóricos (Factors)

Factors são usados para representar dados categóricos, como gênero, níveis educacionais ou regiões geográficas.

01

Criação do Factor

```
genero <- factor(c("feminino",
"masculino", "feminino", "masculino"))
```

02

Tipo Subjacente

```
typeof(genero)
## [1] "integer"
```

03

Estrutura Completa

```
attributes(genero)
## $levels
## [1] "feminino" "masculino"
## $class
## [1] "factor"
```

04

Valores Internos

```
unclass(genero)
## [1] 1 2 1 2
```

Categorias são mapeadas para números

Conversões Entre Tipos

R permite converter entre diferentes tipos de dados usando funções `as.*`. Algumas conversões são automáticas.

Lógico → Numérico

```
sum(c(TRUE, TRUE, FALSE,  
      FALSE))  
## [1] 2
```

```
as.numeric(FALSE)  
## [1] 0
```

`TRUE` = 1, `FALSE` = 0

Numérico → Caractere

```
as.character(1)  
## [1] "1"
```

Números viram texto entre aspas

Numérico → Lógico

```
as.logical(1)  
## [1] TRUE
```

0 = `FALSE`, outros = `TRUE`

Saiba mais: rstudio-education.github.io/hopr/r-objects.html#coercion

Data Frames: Tabelas de Dados

Data frames são estruturas fundamentais para análise de dados, combinando vetores de igual comprimento em uma tabela.

```
df <- data.frame(  
  face = c("ás", "dois", "quatro"),  
  naipe = c("ouros", "copas", "paus"),  
  valor = c(1, 2, 4)  
)  
df  
##   face naipe valor  
## 1   ás  ouros    1  
## 2  dois  copas    2  
## 3 quatro  paus    4
```

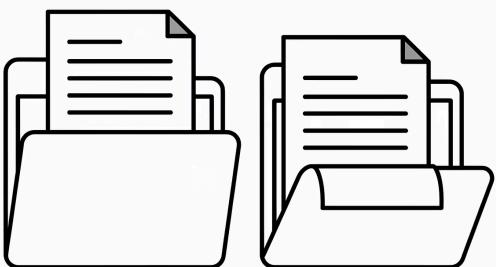
Cada coluna pode ter um tipo diferente:

- **face**: caractere
- **naipe**: caractere
- **valor**: numérico

Perfeito para representar dados tabulares!

Salvando e Lendo Arquivos

R permite exportar e importar dados facilmente, facilitando o compartilhamento e a persistência de resultados.



Salvar CSV

```
write.csv(df,  
          file = "cartas.csv",  
          row.names = FALSE,  
          quote = FALSE)
```

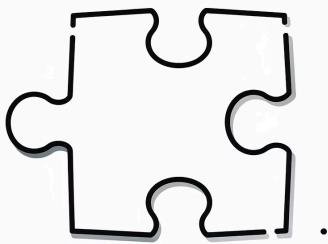
Exporta o data frame para arquivo CSV

Ler CSV

```
cartas <- read.csv("cartas.csv")
```

Importa os dados de volta para R

- Boas Práticas:** Use `row.names = FALSE` para evitar nomes de linhas desnecessários e `quote = FALSE` para texto sem aspas quando apropriado.



⚠ VALORES ESPECIAIS

Valores Ausentes e Especiais em R

Em dados reais, nem toda informação está disponível. O R utiliza valores especiais para representar essas situações de forma apropriada.

	NA (Not Available) Representa um valor ausente ou não disponível nos dados Exemplo: idade não informada em um questionário
	NaN (Not a Number) Resultado numérico indefinido matematicamente Exemplo: divisão de zero por zero (0/0)
	NULL Representa ausência de objeto ou valor nulo Exemplo: variável não inicializada ou resultado vazio

Ideia central: Nem todo erro é um erro de programação. Às vezes, é simplesmente um dado ausente que precisa ser tratado adequadamente.

Lidando com NA em Cálculos

Operações com NA produzem NA. Funções costumam oferecer opções para ignorar valores ausentes durante os cálculos.

Problema: NA contamina resultados

```
x <- c(1, 2, NA, 4)
```

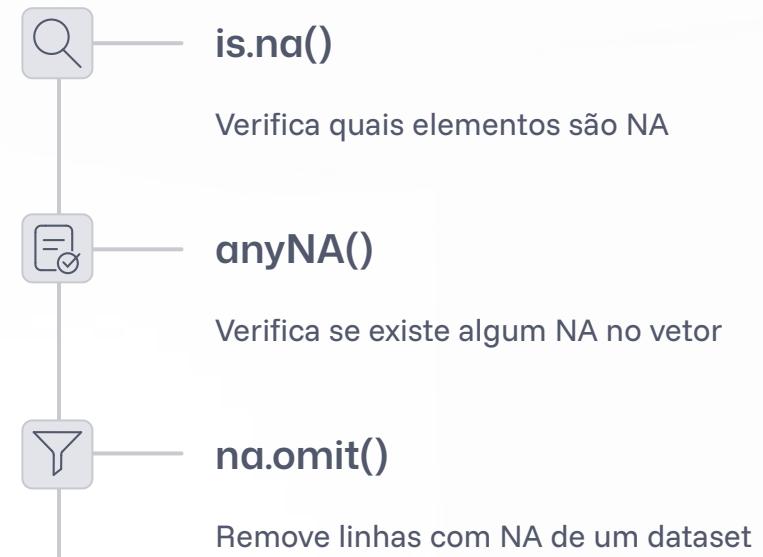
```
mean(x)  
# Resultado: NA
```

O valor ausente impede o cálculo da média

Solução: remover NA

```
mean(x, na.rm = TRUE)  
# Resultado: 2.333333
```

O parâmetro na.rm remove os NA antes do cálculo



Referências



Hands-On
Programming
with R

WRITE YOUR OWN FUNCTIONS AND SIMULATIONS

Hands-on Programming

Aprenda R criando suas
próprias funções e simulações

[https://rstudio-
education.github.io/hopr/index.html](https://rstudio-education.github.io/hopr/index.html)

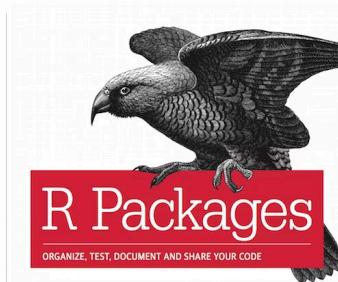


R Graphics
Cookbook

R Graphics Cookbook

Domine visualizações de dados
em R

<https://r-graphics.org>



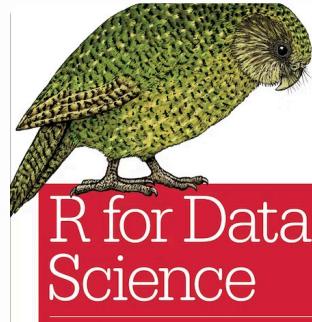
R Packages

ORGANIZE, TEST, DOCUMENT AND SHARE YOUR CODE

R Packages

Desenvolva seus próprios
pacotes R

<https://r-pkgs.org/index.html>



R for Data
Science

VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA

R for Data Science

Guia completo para ciência de
dados

<https://r4ds.had.co.nz>