

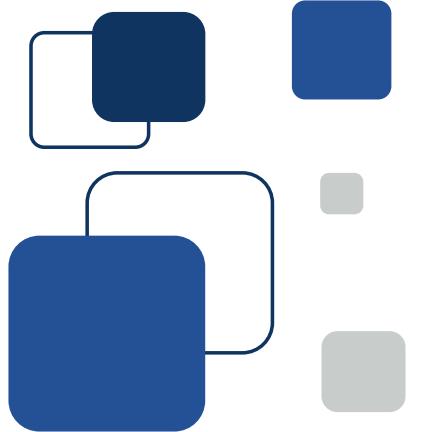
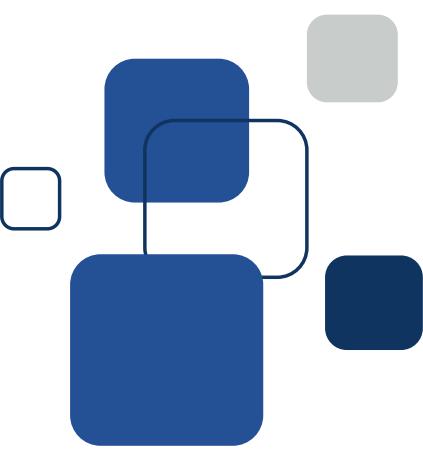


R Project for Statistical Computing

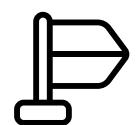
LINGUAGEM R

UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE | AGOSTO 2025
COMPILEDORES E PARADIGMAS DE PROGRAMAÇÃO

Sumário

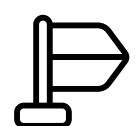
- 
- 
- 01 História da Linguagem R
 - 02 Paradigmas Suportados pela R
 - 03 Aplicações da R
 - 04 Shiny (Framework)
 - 05 Ambientes de Desenvolvimento
 - 06 Hello World!
 - 07 Conslusão

História da R



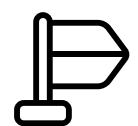
Criada em 1993–1995 por Ross Ihaka e Robert Gentleman

Estatísticos da Universidade de Auckland, Nova Zelândia.



Inspirada na linguagem S

Combinando simplicidade de uso e recursos avançados para análise de dados.

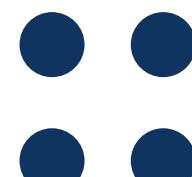


Foco inicial

Análise estatística e computação gráfica de qualidade. Projetada para estatística acadêmica e científica



Ross Ihaka
Robert Gentleman



Paradigmas Suportados

A linguagem R suporta principalmente os seguintes paradigmas de programação:



FUNCIONAL



ORIENTADA A OBJETOS



INTERPRETADA



PROCEDURAL

Aplicações da R



Estatística

R nasceu voltada para estatística e ainda hoje é referência em cálculos, modelagem e testes. Muito usado em pesquisas acadêmicas e na indústria.



Ciência de Dados

R é referência em ciência de dados, com pacotes para mineração, limpeza, exploração e modelagem de grandes volumes de informação.



Bioinformática

Usado em estudos biológicos e médicos, como análise genética e bioestatística, sendo essencial em pesquisas de saúde.

Aplicações da R



Finanças

Aplicado em previsões de mercado, análise de risco e séries temporais, apoiando bancos e empresas de investimento.



Visualização de Dados

Com bibliotecas como ggplot2, R cria gráficos avançados e interativos, tornando dados complexos mais claros e acessíveis.



Relatórios Interativos

Com Shiny, é possível criar dashboards e apps web para compartilhar análises de forma dinâmica e intuitiva.

Shiny



O que é o Shiny?

O Shiny é um framework do R que permite criar aplicações web interativas de forma simples, sem precisar saber HTML, CSS ou JavaScript.



Com o Shiny, você pode:

- Construir dashboards e relatórios dinâmicos;
- Criar interfaces gráficas para análises estatísticas;
- Compartilhar modelos e resultados de forma interativa;
- Integrar gráficos, tabelas e filtros em tempo real.

O Shiny transforma códigos e análises em R em aplicações web completas, muito usado em pesquisa, ciência de dados e até em empresas para relatórios executivos.



Ambientes de Programação em R



Bibliotecas / Frameworks

- ggplot2 → gráficos e visualização
- dplyr → manipulação de dados
- Shiny → aplicações web interativas

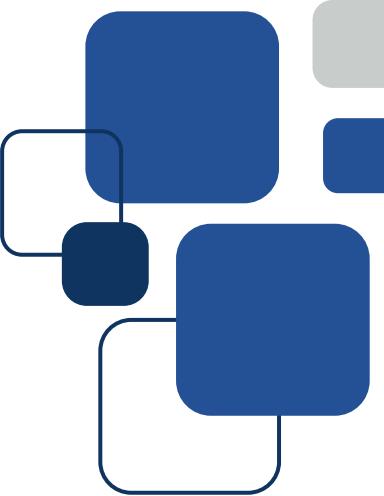


IDEs

- RStudio (mais popular)
- VS Code (com extensão de R)
- Jupyter Notebook (ambiente interativo)



O RStudio mudou de nome para Posit, para refletir a expansão da empresa para além do R, incluindo o Python e outras ferramentas de código aberto



 **HelloWorld.ipynb**  

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

Comandos + Código + Texto ▶ Executar tudo



```
[4] print("Hello, World!")  
[1] "Hello, World!"
```

Hello World!

Testamos um código simples em R que imprime "Hello, World!" no console. Esse exercício demonstra como R funciona de forma interpretada, permitindo que comandos sejam executados linha a linha.

Observações Importantes



O R executa o código linha a linha, permitindo resultados imediatos e gráficos interativos sem complicações.

Tipos de Dados



Primitivos

- Primitivos: numeric (double 64-bit), integer (32-bit), logical, character, complex



Compostos

- Compostos: vector, list, matrix, data.frame, factor
- Vetores dinâmicos; indexação 1-based (inteiros, lógicos, nomes)



Formato de Implementação dos Tipos

- Vetores: armazenamento contíguo por tipo (operações vetorizadas rápidas)
- Listas: heterogêneas, contêm referências a elementos (não contíguos)
- Índices: 1-based (inteiros, lógicos ou nomes)

```
is.double(3.14)  
is.integer(2L)  
v["nome"] <- 5
```

- numeric → ponto flutuante (IEEE-754, 64 bits)
- integer → 32 bits (sufixo L)
- logical / character → tipos específicos; character usa pool de strings/encodings (UTF-8)

Uso de Ponteiros e Referências

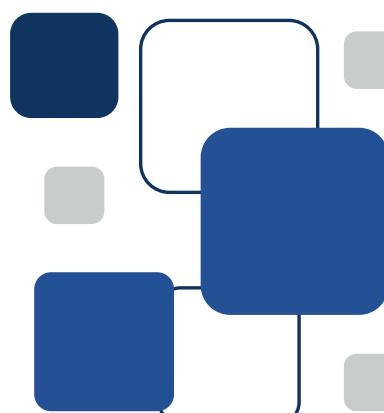
- Não há ponteiros explícitos (sem aritmética de ponteiros)
- Cópia por valor lógica (copy-on-modify) — eficiente até modificação
- Referência explícita: environment, R6, data.table (modificação por referência), externalptr (C)

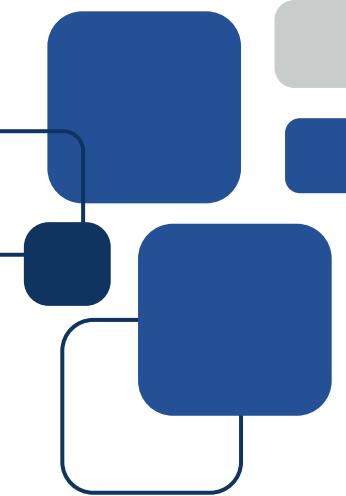
```
env <- new.env(); env$a <- 1
modify_env <- function(e) { e$a <- 2 }
modify_env(env); env$a
```



Palavras-chave e Reservadas

- **Palavras-chave de controle:** if, else, repeat, while, for, in, function, next, break
- **Constantes especiais / valores NA/Inf:** TRUE, FALSE, T, F, NULL, NA, NaN, Inf
- **NAs específicos:** NA_integer_, NA_real_, NA_character_, NA_complex_
- **Ellipsis / argumentos:** ...
- **Operadores / símbolos importantes:** <-, <<-, ->, ->>, =, :, ::, ..., \$, @, ~
- **Evitar sobrescrever funções base comuns:** c, sum, mean, data.frame, list, print, length, as.character, read.csv





Variáveis: nomes, vinculação, tempo de vida e escopo



- Nomes: case-sensitive; permitem letras, números, . e _; não iniciar com número
- Vinculação: tipagem dinâmica (binding em tempo de execução)
- Tempo de vida: enquanto existir no ambiente (garbage collection)
- Escopo: lexical (funções têm escopo léxico; ambientes definem visibilidade)



Variáveis

Cada variável é um nome que referencia um objeto; o valor pode mudar sem alterar o nome.

Expressões: associatividade, precedência e sobrecarga



Expressões

- Precedência: $\wedge > * / > + - >$ comparadores $> \& |$
- Associatividade: usual matemática (ex.: 2^3^2 associatividade à direita)
- Sobrecarga: operadores são genéricos (Ops, S3/S4 methods) e vetorizados

```
a <- 2 + 3 * 4    # 14
b <- 2^3^2          #  $2^{\wedge}(3^{\wedge}2) = 2^9$ 
v <- c(1,2) + c(3,4) # vetor somado
```

Comandos Condicionais (tipos e implementação)



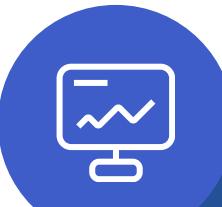
Comandos

- if, if ... else — avalia condição escalar (use ifelse para vetores)
- switch para seleção por valor/índice
- ifelse para vetores (element-wise)

```
x <- 10
if (x > 5) { print("maior") } else { print("menor/igual") }

v <- c(TRUE, FALSE, TRUE)
ifelse(v, "T", "F")
```

Comandos de Repetição (tipos e implementação)



Comandos

- Loops: for, while, repeat (+ break, next)
- Alternativas vetorizadas/funcionais: apply, lapply, sapply, purrr
- repeat = loop infinito até break

```
for (i in 1:3) print(i)
```

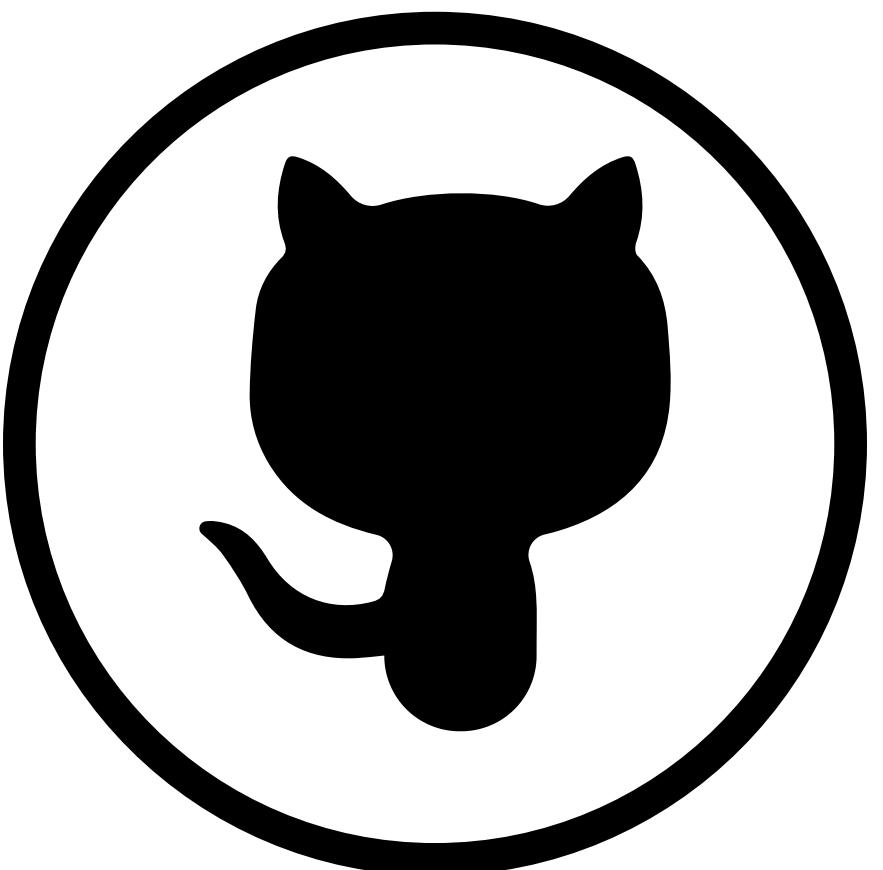
```
count <- 1
while (count <= 3) { print(count); count <- count + 1 }
```

```
repeat { print("x"); break } # executa ao menos 1 vez
```

Acesse os exemplos completos no GitHub



aprendendo-R (Repositório)





R Project for Statistical Computing

**OBRIGADO PELA
ATENÇÃO!**