

MAC 113 — Introdução à Ciência da Computação

Aula 4

Nelson Lago

1º/2025



Previously on MAC 113...

Programando

- 1 algoritmo vs implementação
- 2 entrada de dados → processamento → resultado
 - ▶ Mostra o resultado para o usuário
 - ▶ **Utiliza o resultado como dado para fazer outra coisa**
- 3 Existem *tipos de dados* diferentes em R (*integer*, *numeric*, *character*, *logical*...)
- 4 Expressões são coisas que têm um *valor* (de algum *tipo*)
 - ▶ E podem ser combinadas ou utilizadas como partes de outras expressões



2 + 3 + 7 (integer)

2 > 3 && 5 > 4 (logical)

Expressões em R

- A maioria das coisas em R são *expressões*
 - ▶ (expressões são coisas que têm um *valor*)
 - ▶ Exemplo: `2 > 3`
- AAAAAHHHH!!!!!!!
- Qual o resultado dessa heresia?

```
cat(2 > 3, "\n")
```

```
FALSE
```

Ufa!

Tipos booleanos

- O que exatamente são **TRUE** e **FALSE**?
- São um outro *tipo* de dado (como inteiros, *strings* e números de ponto flutuante)
- Esse tipo se chama *booleano* (em homenagem a George Boole)
 - Em R, o nome usado é *logical*
- Tipos booleanos só podem ter dois valores: **TRUE** e **FALSE**

Tipos

```
cat(class(2L), "\n")  
cat(class(FALSE), "\n")  
cat(class(2.0), "\n")  
cat(class("Olá"), "\n")
```

integer
logical
numeric
character

Operadores relacionais

operador	descrição
$==$	igualdade
$!=$	desigualdade
$>$	maior
$>=$	maior ou igual ($\geq$)
$<$	menor
$<=$	menor ou igual ($\leq$)

Operadores relacionais em R

Programando em R

- A maioria das coisas em R são *expressões*
 - ▶ (expressões são coisas que têm um *valor*)
 - ▶ Exemplo: 47
 - ▶ Exemplo: $2 + 3$
 - ▶ Exemplo: $2 > 3$
- Expressões podem ser combinadas ou utilizadas como partes de outras expressões
 - ▶ Exemplo: $\frac{2+3+7}{6}$
 - ▶ Exemplo: $2 + 3 + 7 < 4$
 - ▶ Exemplo: $2 + 2 == 2 * 2$



Operadores relacionais

```
cat(2 + 3 + 7 < 4, 2 + 2 == 2 * 2, "\n")
```

```
FALSE TRUE
```

**Os operandos são inteiros, mas o
resultado da expressão é um booleano**

Expressões lógicas

- A maioria das coisas em R são *expressões*
 - ▶ (expressões são coisas que têm um *valor*)
 - ▶ Exemplo: 47
 - ▶ Exemplo: $2 + 3$
 - ▶ Exemplo: $2 > 3$
- Expressões podem ser combinadas ou utilizadas como partes de outras expressões
 - ▶ Exemplo: $2 + 3 + 7 < 4$
 - ▶ Exemplo: $2 > 3 \ \&\& \ 5 > 4$



Expressões lógicas

```
cat(2 > 3, "\n")
```

```
cat(5 > 4, "\n")
```

```
cat(2 > 3 && 5 > 4, "\n")
```

```
cat(2 > 3 || 5 > 4, "\n")
```

FALSE

TRUE

FALSE

TRUE

Precedência de operadores

operador	descrição
!	negação lógica (“inverte o sinal”)
&&	E lógico (só é verdadeiro se ambos os operandos são verdadeiros)
	OU lógico (só é verdadeiro se ao menos um dos operandos é verdadeiro)

Precedência dos operadores lógicos em R (da maior para a menor)

Operadores lógicos

A && B

	A = TRUE	A = FALSE
B = TRUE	TRUE	FALSE
B = FALSE	FALSE	FALSE

Tabela verdade do operador &&

Operadores lógicos

A || B

	A = TRUE	A = FALSE
B = TRUE	TRUE	TRUE
B = FALSE	TRUE	FALSE

Tabela verdade do operador ||

Expressões lógicas e tipos

Só vou à praia se:

- A temperatura no fim-de-semana for maior que 25°C
- Eu tiver pelo menos R\$100
- A minha nota na prova for A ou B

```
temp <- 29L
saldo <- 100.39
nota <- "B"
cat(temp > 25 && saldo >= 100 && (nota == "A" || nota == "B"), "\n")
```

```
TRUE
```

integer

character

numeric

logical

Nomes (funções)

- *Nomes* permitem que pensemos mais sobre o problema a ser resolvido e menos sobre as idiossincrasias do computador
- Variáveis são **expressões** (têm um valor)
 - ▶ `nome <- "Fulano"`
`idade <- 27`
- Nomes também podem ser usados para representar “ações” (**como calcular** o valor de uma expressão)
 - ▶ Por exemplo, `sqrt()` é o **nome** que usamos para representar o cálculo da raiz quadrada
- Funções recebem um valor, realizam algum processamento e devolvem outro valor correspondente

Exercício – área do triângulo

A área de um triângulo de lados a, b, c é dada por

$$\sqrt{s \cdot (s - a) \cdot (s - b) \cdot (s - c)}, \text{ onde } s = \frac{a+b+c}{2}$$

Modifique o programa abaixo para obter os valores de a, b, c do usuário e calcular a área usando uma função

```
area <- function(a, b, c) {  
  s <- (a + b + c) / 2  
  return(sqrt(s * (s-a) * (s-b) * (s-c)))  
}  
main <- function() {  
  a <- as.integer(readline("Digite o primeiro lado: "))  
  b <- as.integer(readline("Digite o segundo lado: "))  
  c <- as.integer(readline("Digite o terceiro lado: "))  
  cat("A área do triângulo dado é", area(a, b, c), "\n")  
}  
main()
```

Comandos essenciais vistos até agora

- `x <- 5` → atribui o valor 5 ao nome `x`
- `cat(blah, "\n")` → imprime o valor da variável `blah`
- `cat("blah", "\n")` → imprime o texto `blah`
- `x <- readline()` → lê o que o usuário digita e atribui ao nome `x`
 - ▶ `cat("Qual o seu nome?", "\n")`
`nome <- readline()`
 - ▶ Equivale a:
`nome <- readline("Qual o seu nome? ")`
- `celsius <- function(f) { return(5 * (f - 32) / 9) }`
→ cria uma função com o nome `celsius()`
- `temp <- celsius(96)` → expressão usando a função `celsius()`

Formatação de strings

```
library(glue)
v <- as.numeric(readline("Velocidade do vento (Km/h): "))
t <- as.numeric(readline("Temperatura (°C): "))
u <- as.numeric(readline("Umidade relativa (%): ")) / 100
sensação <- t + 2.015 * u * exp(17.27*t/(237.7+t)) - 0.194*v - 4
cat(glue("Sensação térmica: {round(sensação, 1)}°C"), "\n")
```

Velocidade do vento (Km/h): 10

Temperatura (°C): 25

Umidade (%): 50

Sensação térmica: 24.3°C

- O resultado de `glue()` é uma *expressão*
- O valor da expressão é uma nova string

and now for something completely different

Execução condicional

```
library(glue)
v <- as.numeric(readline("Velocidade do vento (Km/h): "))
t <- as.numeric(readline("Temperatura (°C): "))
u <- as.numeric(readline("Umidade relativa (%): ")) / 100
sensação <- t + 2.015 * u * exp(17.27*t/(237.7+t)) - 0.194*v - 4
cat(glue("Sensação térmica: {round(sensação, 1)}°C"), "\n")
if (sensação < 18) {
  cat("Leva um casquinho!", "\n")
}
```

Execução condicional 1/2

```
n <- as.integer(readline("Digite um número natural: "))  
  
if (n %% 2 == 0) {  
  cat("O número", n, "é par!", "\n")  
}
```

- `n %% 2 == 0` → **TRUE** ou **FALSE**

- ▶ Embora possamos ler “se condição”, na verdade R faz “se o valor da expressão é verdadeiro (**TRUE**)”
- ▶ É **como se** R executasse `if` (condição == **TRUE**)

Execução condicional 2/2

```
n <- as.integer(readline("Digite um número natural: "))

if (n %% 2 == 0) {
  ➡ cat("O número", n, "é par!", "\n")
}

cat("Ahazei!", "\n")
```

- Ele sempre imprime “Ahazei!” ou só quando o número é par?
- O bloco a ser executado ou não é definido por { e }
 - ▶ Novamente, a indentação é uma convenção, mas na prática é essencial

Execução condicional – else

```
n <- as.integer(readline("Digite um número natural: "))
if (n %% 2 == 0) {
  cat("O número", n, "é par!", "\n")
} else {
  cat("O número", n, "é ímpar!", "\n")
}
cat("Ahazei!", "\n")
```

- Há dois “lados” do condicional

- ▶ Sem uma variável (“n”), o programa sempre executaria o mesmo “lado”

- O estado da variável só importa no momento do teste

- ▶ Se ela mudar em seguida, não afeta o condicional

- Com else, os “lados” são mutuamente excludentes

- ▶ Um e apenas um deles é executado

Exercício — ordem crescente

Dados três números, verifique se eles estão em ordem crescente (considere que números iguais estão em ordem crescente)

```
a <- as.integer(readline("Digite o primeiro número: "))
b <- as.integer(readline("Digite o segundo número: "))
c <- as.integer(readline("Digite o terceiro número: "))
if (a <= b && b <= c) {
  cat("Os números estão em ordem crescente", "\n")
} else {
  cat("Os números não estão em ordem crescente", "\n")
}
```

Exercício — triângulo retângulo

Dados três números naturais *em ordem crescente*, verifique se eles formam os lados de um triângulo retângulo

```
a <- as.integer(readline("Digite o primeiro número: "))
b <- as.integer(readline("Digite o segundo número: "))
c <- as.integer(readline("Digite o terceiro número: "))
if (c**2 == a**2 + b**2) {
  cat("Os números formam um triângulo retângulo", "\n")
} else {
  cat("Os números não formam um triângulo retângulo", "\n")
}
```

Exercício

```
# Calcula raízes da equação de segundo grau
a <- -1
b <- 3
c <- 4
delta <- b**2 - 4 * a * c # Não pode ser menor que zero!
if (delta >= 0) {
  raiz1 <- (-1 * b + sqrt(delta)) / 2 * a
  raiz2 <- (-1 * b - sqrt(delta)) / 2 * a
  cat("As raízes são", raiz1, "e", raiz2, "\n")
} else {
  cat("A equação não tem raízes reais", "\n")
}
```

- Melhore o programa acima para verificar se Δ é maior ou igual a zero

Exercício — sistema de *login*

Crie um sistema que lê o UID (*login*) e a senha do usuário e, se os dados estiverem corretos, escreve “Bem-vindo!”; caso contrário, o sistema escreve “Login ou senha incorreto”. Os dados de acesso são “ali babá” e “abre-te, sésamo”.

```
uid <- readline("username: ")
senha <- readline("senha: ")
if (uid == "ali babá" && senha == "abre-te, sésamo") {
  cat("Bem-vindo!", "\n")
} else {
  cat("Login ou senha incorreto", "\n")
}
```

Exercício — sistema de *login*

Imagine um sistema de *login* com três usuários:

- **Alan Turing** — UID **turing**, senha **tmachine**
- **Ada Lovelace** — UID **llace**, senha **anengine**
- **Grace Hopper** — UID **hopper**, senha **business**

Crie um sistema que lê o UID (*login*) e a senha do usuário e, se os dados estiverem corretos, escreve “Bem-vindo!”; caso contrário, o sistema escreve “Login ou senha incorreto”.

Exercício — sistema de *login*

```
uid <- readline("username: ")
senha <- readline("senha: ")
if (uid == "turing" && senha == "tmachine"
    || uid == "llace" && senha == "anengine"
    || uid == "hopper" && senha == "business") {
  cat("Bem-vindo!", "\n")
} else {
  cat("Login ou senha incorreto", "\n")
}
```

Blocos

- Muitas vezes precisamos indicar que uma sequência de linhas é um “bloco” (definição de funções, condicionais...)
 - ▶ Ou seja, é preciso indicar qual a primeira e qual a última linhas do conjunto
- Em R, o bloco é indicado pelos caracteres { e }
 - ▶ Mas **sempre** usamos também a **indentação**!
- O que nos impede de termos um bloco dentro do outro? **NADA!**

```
if (sensação < 18) {  
  cat("Leva um casquinho!", "\n")  
  if (sensação < 10) {  
    cat("E um cachecol!", "\n")  
  }  
} else {  
  cat("Leva protetor solar!", "\n")  
}
```

Exercício — cálculo da nota 1/3

Faça um programa que recebe as notas de um aluno desta disciplina e calcula sua média final de acordo com os critérios definidos no eDisciplinas

```
p1 <- as.numeric(readline("Digite a nota da prova 1 "))
p2 <- as.numeric(readline("Digite a nota da prova 2 "))
ep1 <- as.numeric(readline("Digite a nota do EP 1 "))
ep2 <- as.numeric(readline("Digite a nota do EP 2 "))
mp <- (p1 + 2*p2)/3
mep <- (ep1 + 2*ep2)/3
if (mp >= 5 && mep >= 5) {
  mf <- (mep + 2*mp)/3
} else {
  if (mp < mep) {
    mf <- mp
  } else {
    mf <- mep
  }
}
cat("A média final é", mf, "\n")
```