Aula 02 - Tipos de Dados e Estruturas Basicas - Curso R para iniciantes

Adhemar Ranciaro Neto

Aula 02 – Tipos de Dados e Estruturas Básicas

Nesta aula, vamos aprender os principais tipos de dados do R e como usá-los em vetores, fatores e outras estruturas úteis para análise de dados.

Tipos básicos de dados no R

```r
# Numérico
idade <- 35
class(idade)

# Caractere (texto)
nome <- "Carlos"
class(nome)

# Lógico (TRUE ou FALSE)
ativo <- TRUE
class(ativo)
```

Vetores

Vetores são conjuntos de elementos do mesmo tipo.

```r
nomes <- c("Maria", "João", "Ana")
idades <- c(23, 35, 31, 93, 22, 15, 19, 65)

# Acessar elementos
nomes[1]
idades[3]
idades[2:3]
idades[3:6]
```

Operações com vetores

```r

# soma de vetores

salarios <- c(2500, 3200, 2800)
```

```r
bonus <- c(200, 300, 250)

salario_total <- salarios + bonus
salario_total

# Subtração entre vetores numéricos

idades <- c(23, 35, 31)
ajuste <- c(1, 0, -2)
novas_idades <- idades - ajuste
novas_idades

# multiplicação elemento a elemento de vetor

nota<-c(7,8,3)
peso<-c(0.2,0.5,0.3)

nota_corrigida<-nota*peso
nota_final=nota_corrigida[1]+nota_corrigida[2]+nota_corrigida[3]

# soma dos elementos de um vetor

nota_final_2=sum(nota_corrigida)

# Operações lógicas com vetores (Ex: verificar um critério)

idades <- c(23, 35, 31)
idades > 30       # Retorna TRUE para idades maiores que 30

check_idade <- (idades > 30)
print(check_idade)
```

Fatores

Fatores são usados para representar variáveis categóricas.

```r
setores <- c("Financeiro", "RH", "Vendas", "Financeiro")
fator_setores <- factor(setores)

fator_setores
levels(fator_setores)

fator_setor_ordenado <- factor(setores, levels = c("RH", "Vendas", "Financeiro"))
levels(fator_setor_ordenado)

sexo <- factor(c("M", "F", "F", "M"))
class(sexo)
levels(sexo)

sexo_L <- factor(c("M", "F", "F", "M"), levels = c("M","F"))
levels(sexo_L)
```

```r
curso <-c("Economia", "Economia", "Contábeis", "Administração", "Artes Cênicas", "Administração")

curso_fator <- factor(curso, levels=c("Economia","Contábeis","Artes Cênicas", "Administração"))

levels(curso_fator)

nivel_educacional <- factor(
  c("Médio", "Superior", "Médio", "Fundamental"),
  levels = c("Fundamental", "Médio", "Superior"),
  ordered = TRUE)
```

Listas

Listas armazenam elementos de diferentes tipos.

Criando listas

```r
# formas igualmente válidas de escrever um comando extenso

dados_i <- list(nome = "Carlos", idade = 35, ativo = TRUE, notas = c(8.5, 7.9, 9.0))

dados <- list(nome = "Carlos",
              idade = 35,
              ativo = TRUE,
              notas = c(8.5, 7.9, 9.0)
              )


dados_i$idade # Acessa o valor da chave idade
dados$idade

dados$nome     # Acessa elemento pelo nome
dados[[2]]     # Acessa pelo índice
dados[[4]]
```

```r
# Lista com elementos de tipos diferentes
info <- list(
  nome = "Carlos",
  idade = 35,
  ativo = TRUE,
  notas = c(8.5, 7.9, 9.0)
)

info
```

Acessando elementos da lista

```r
info$nome          # Pelo nome
info[["idade"]]    # Pelo nome entre colchetes duplos
info[[4]]          # Pelo índice

info[1]            # Retorna uma sublista
info[[1]]          # Retorna o conteúdo direto (string "Carlos")
```

Verificando estrutura

```r
str(info)       # Estrutura da lista
length(info)    # Número de elementos
names(info)     # Nomes dos elementos
```

Adicionando elementos

```r
info$email <- "carlos@email.com"
info
```

Ou:

```r
info[["cidade"]] <- "Maceió"
```

Removendo elementos

```r
info$cidade <- NULL
info
```

Modificando elementos

```r
info$idade <- 36
info$notas[2] <- 8.3
```

Listas dentro de listas (listas aninhadas)

```r
aluno <- list(
  nome = "Ana",
  dados = list(
    idade = 22,
    curso = "Administração"
  ),
  notas = c(9.0, 8.5, 10)
)

# Acessando dados aninhados
```

```r
aluno$dados$curso
aluno[["dados"]][["idade"]]
```

Aplicando funções com `lapply()` e `sapply()`

```r
# Criando uma lista de vetores
numeros <- list(a = 1:5, b = 6:10, c = 11:15)

# Soma de cada vetor
lapply(numeros, sum)    # Retorna uma lista
sapply(numeros, sum)    # Retorna um vetor (mais compacto). Se não der, retorna uma lista.
```

Convertendo lista para data frame (quando possível)

```r
dados <- list(
  nome = c("Ana", "Bruno", "Carlos"),
  idade = c(21, 25, 24),
  ativo = c(TRUE, TRUE, FALSE)
)

df <- as.data.frame(dados)
df
```

Exemplo prático com lista de alunos

```r
alunos <- list(
  aluno1 = list(nome = "Ana", notas = c(9, 8, 10)),
  aluno2 = list(nome = "Bruno", notas = c(7, 6, 8)),
  aluno3 = list(nome = "Carla", notas = c(10, 9, 9.5))
)

# Média das notas de cada aluno
sapply(alunos, function(x) mean(x$notas))
```

Matrizes

Matrizes são estruturas bidimensionais (linhas x colunas) com elementos do mesmo tipo.

Criando Matrizes

```r
# Criar uma matriz 2x3 preenchida por colunas
A <- matrix(1:6, nrow = 2, ncol = 3)
A

# Criar preenchendo por linha
B <- matrix(1:6, nrow = 2, byrow = TRUE)
```

```r
B

notas <- matrix(
  c(8, 7, 9,
    6, 8, 5,
    9, 9, 10),
  nrow = 3,
  byrow = TRUE
)
```

Indexação de Matrizes

```r
A[1, 2]     # Elemento da 1ª linha, 2ª coluna
A[2, ]      # Toda a 2ª linha
A[, 3]      # Toda a 3ª coluna
```

Operações entre Matrizes

```r
M1 <- matrix(c(1, 2, 3, 4), nrow = 2)
M2 <- matrix(c(5, 6, 7, 8), nrow = 2)

M1 + M2     # Soma elemento a elemento
M2 - M1     # Subtração
M1 * M2     # Multiplicação elemento a elemento
```

Multiplicação Matricial (%*%)

```r
A <- matrix(c(1, 2, 3, 4), nrow = 2)     # 2x2
B <- matrix(c(2, 0, 1, 3), nrow = 2)     # 2x2

A %*% B     # Multiplicação matricial padrão (produto interno)
```

Transposta da Matriz

```r
A <- matrix(1:6, nrow = 2)
t(A)       # Transposta (linhas viram colunas)
```

Determinante

```r
M <- matrix(c(2, 3, 1, 4), nrow = 2)
det(M)
```

Inversa da Matriz

```r
M <- matrix(c(2, 3, 1, 4), nrow = 2)
solve(M)       # Retorna a inversa da matriz
```

A matriz precisa ser **quadrada** e com **determinante diferente de zero**.

Operações por Linha ou Coluna com `apply()`

```r
mat <- matrix(1:9, nrow = 3)

apply(mat, 1, sum)   # Soma por linha (1 = linha)
apply(mat, 2, mean)  # Média por coluna (2 = coluna)
```

Exemplo Integrado

```r
notas <- matrix(
  c(8, 7, 9,
    6, 8, 5,
    9, 9, 10),
  nrow = 3,
  byrow = TRUE
)

rownames(notas) <- c("Aluno1", "Aluno2", "Aluno3")
colnames(notas) <- c("Prova1", "Prova2", "Prova3")

notas

# Média por aluno
apply(notas, 1, mean)

# Média por prova
apply(notas, 2, mean)
```

Data Frames (introdução)

Data frames são uma das estruturas mais usadas no R para representar dados em formato de tabela.

Criando um `data.frame`

```r
df <- data.frame(
  nome = c("Maria", "João", "Ana"),
  idade = c(23, 35, 31),
  ativo = c(TRUE, TRUE, FALSE)
)

str(df)      # Mostra a estrutura do data frame
```

```r
df$idade      # Acessa a coluna "idade"
df[1, 2]      # Acessa o valor da primeira linha, segunda coluna
```

```r
dados <- data.frame(
  nome = c("Ana", "Bruno", "Carlos", "Daniela"),
  idade = c(22, 25, 24, 21),
  curso = c("ADM", "ECON", "CONT", "ADM"),
  ativo = c(TRUE, TRUE, FALSE, TRUE)
)

dados
```

Acessando colunas, linhas e células

```r
dados$idade          # Coluna 'idade'
dados[2, ]           # Segunda linha
dados[, 3]           # Terceira coluna
dados[1, 2]          # Célula da 1ª linha, 2ª coluna
```

Estrutura e resumo

```r
str(dados)           # Estrutura do data frame
summary(dados)       # Resumo estatístico
nrow(dados)          # Número de linhas
ncol(dados)          # Número de colunas
names(dados)         # Nomes das colunas
```

Adicionando colunas e linhas

```r
# Adicionando uma coluna
dados$nota <- c(8.5, 7.2, 9.0, 6.8)

# Adicionando uma linha
nova_linha <- data.frame(
  nome = "Eduardo",
  idade = 23,
  curso = "ECON",
  ativo = TRUE,
  nota = 8.0
)

dados <- rbind(dados, nova_linha)
```

Removendo colunas ou linhas

```r
dados$ativo <- NULL         # Remove a coluna 'ativo'
dados <- dados[-2, ]        # Remove a 2ª linha
```

Filtrando linhas com condições

```r
dados[dados$idade > 22, ]                  # Idade maior que 22
dados[dados$curso == "ADM", ]              # Alunos de Administração
dados[dados$nota >= 8 & dados$curso == "ECON", ]  # Notas boas na Economia
```

Ordenando um `data.frame`

```r
# Ordenar por idade crescente
dados[order(dados$idade), ]

# Ordenar por nota decrescente
dados[order(-dados$nota), ]
```

`subset()` – Filtro com mais clareza

```r
subset(dados, curso == "ECON" & nota >= 8)
```

Convertendo variáveis para fator

```r
dados$curso <- as.factor(dados$curso)
str(dados)
```

Estatísticas por grupo com `tapply()`

```r
# Média de notas por curso
tapply(dados$nota, dados$curso, mean)
```

Exemplo de aplicação prática

```r
# Média de idade dos alunos ativos
media_idade_ativos <- mean(dados$idade[dados$ativo == TRUE])
print(media_idade_ativos)
```

Juntando Data Frames com `merge()`

A função `merge()` permite **combinar dois data frames** com base em uma ou mais colunas em comum (como um "vlookup" no Excel).

Exemplo:

```r
alunos <- data.frame(
  id = c(1, 2, 3),
  nome = c("Ana", "Bruno", "Carlos")
)

notas <- data.frame(
  id = c(1, 2, 3),
  nota_final = c(8.5, 7.2, 9.0)
)

# Junta os dois data frames pela coluna "id"
dados_completos <- merge(alunos, notas, by = "id")
dados_completos
```

Tipos de merge

Você pode controlar o tipo de junção:

```r
# Inner join (padrão) Junta apenas as linhas em que o valor da coluna usada no by está presente nos doi

df1 <- data.frame(id = c(1, 2, 3), nome = c("Ana", "Bruno", "Carlos"))
df2 <- data.frame(id = c(2, 3, 4), nota = c(8.5, 7.2, 9.0))

merge(df1, df2, by = "id")

# Left join. Mantém todas as linhas de x, e adiciona informações de y se houver correspondência. Se não

merge(df1, df2, by = "id", all.x = TRUE)

# Right join. Mantém todas as linhas de y, e pega as correspondências de x.

merge(df1, df2, by = "id", all.y = TRUE)

# Full outer join. Junta todos os dados de x e y, preenchendo com NA onde não houver correspondência.

merge(df1, df2, by = "id", all = TRUE)
```

Adicionando linhas com `rbind()` (append de data frames)

Para **acrescentar linhas** de um data frame ao outro (desde que tenham as mesmas colunas):

```r
df1 <- data.frame(nome = c("Ana", "Bruno"), idade = c(22, 25))
df2 <- data.frame(nome = c("Carlos", "Daniela"), idade = c(24, 21))

df_total <- rbind(df1, df2)
df_total
```

Tipos Lógicos

No R, o tipo lógico (ou booleano) representa apenas dois valores: TRUE ou FALSE

```
gasto <- 4000
orcamento <- 3500

gasto > orcamento  # TRUE
gasto <= 5000      # TRUE


T <- "texto"  # Isso sobrescreve o valor lógico! Evite!

x <- 10
x > 5      # TRUE
x == 10    # TRUE
x != 7     # TRUE
x < 3      # FALSE
```

Tabela de Operadores Lógicos no R

Operador	Descrição	Exemplo	Resultado
`&`	E (AND)	`TRUE & FALSE`	`FALSE`
`|`	OU (OR)	`TRUE | FALSE`	`TRUE`
`!`	NÃO (NOT)	`!TRUE`	`FALSE`

Exemplos com Operadores Lógicos

`&` – E lógico (AND)

Retorna `TRUE` apenas se **ambos** os valores forem `TRUE`.

```
TRUE & TRUE      # TRUE
TRUE & FALSE     # FALSE
FALSE & FALSE    # FALSE
```

Exemplo com vetor:

```
idades <- c(16, 22, 30, 17)
ingressos <- c(TRUE, TRUE, FALSE, TRUE)

# Ver quem tem 18+ E comprou ingresso
idades >= 18 & ingressos
# Resultado: FALSE  TRUE FALSE FALSE
```

`|` – OU lógico (OR)

Retorna `TRUE` se **pelo menos um** dos valores for `TRUE`.

```
TRUE | FALSE     # TRUE
FALSE | FALSE    # FALSE
TRUE | TRUE      # TRUE
```

Exemplo com vetor:

```
idoso <- c(FALSE, FALSE, TRUE, FALSE)
estudante <- c(TRUE, FALSE, FALSE, TRUE)

# Tem direito a desconto (se for idoso OU estudante)
idoso | estudante
# Resultado: TRUE FALSE TRUE TRUE
```

! – NÃO lógico (NOT)

Inverte o valor lógico: `TRUE` vira `FALSE`, e vice-versa.

```
!TRUE    # FALSE
!FALSE   # TRUE
```

Exemplo com vetor:

```
ativo <- c(TRUE, TRUE, FALSE, TRUE)
!ativo
# Resultado: FALSE FALSE TRUE FALSE
```

Combinações mais complexas

Você pode combinar operadores:

```
idade <- 20
tem_documento <- TRUE

(idade >= 18) & tem_documento   # TRUE
(idade >= 18) & !tem_documento  # FALSE
```

Exemplos com vetores

```
idades <- c(15, 22, 30, 17)

# Quais são maiores de idade?
idades >= 18
# Resultado: FALSE TRUE TRUE FALSE

# Filtro: quais idades são maiores que 20?
idades[idades > 20]
```

Tipos lógicos são essenciais para selecionar dados com condições:

```
dados <- data.frame(
  nome = c("Ana", "Bruno", "Carlos"),
  idade = c(22, 17, 19)
)

# Selecionar apenas maiores de idade
dados[dados$idade >= 18, ]
```

Mini Desafios

1. Crie um vetor com os nomes de 4 cidades.
2. Crie um vetor com o PIB (em milhões) dessas cidades.
3. Calcule o PIB total.
4. Classifique o vetor de cidades como fator.
5. Crie uma lista contendo o cadastro de um tipo de automóvel
6. Crie uma matriz quadrada e calcule sua inversa
7. Crie um dataframe contendo nome idade sexo e altura para 4 pessoas 8, Ordende o dataframe da questão 7 da maior altura para a menor altura.
8. transforme a coluna sexo em fator

Algumas respostas

```r
cidades <- c("Maceió", "Recife", "Salvador", "Natal")
pib <- c(27000, 56000, 62000, 31000)

pib_total <- sum(pib)

fator_cidades <- factor(cidades)
```

Guia Rápido: Verificação e Conversão de Tipos no R

Funções `is.` — Verificam o tipo do objeto

```r
is.numeric(10)              # TRUE
is.character("abc")         # TRUE
is.logical(TRUE)            # TRUE
is.integer(10L)             # TRUE
is.factor(factor("A"))      # TRUE
is.matrix(matrix(1:4, 2))   # TRUE
is.data.frame(mtcars)       # TRUE
is.list(list(a = 1))        # TRUE
```

Funções `as.` — Convertem para outro tipo

```r
as.numeric("10.5")                    # 10.5
as.character(123)                     # "123"
as.logical(0)                         # FALSE
as.integer(10.7)                      # 10
as.factor(c("A", "B", "A"))           # Converte para fator
as.matrix(data.frame(x = 1:3))        # Converte para matriz
as.data.frame(matrix(1:4, ncol = 2))  # Converte para data frame
as.list(c(1, 2, 3))                   # Converte vetor para lista
```

Atenção com conversões inválidas

```r
as.numeric("abc")   # Retorna NA e gera um aviso
```

Dica

Use `is.` para testar o tipo antes de aplicar funções específicas,
e `as.` para converter tipos quando necessário durante análises.

Tarefa para casa

Parte 1 – Exercícios Básicos

1. Crie dois vetores com despesas e receitas mensais de uma empresa por 6 meses.
2. Calcule o lucro mensal e o lucro total da empresa.
3. Classifique os meses como fator e associe-os aos valores de lucro.
4. Crie um vetor com 5 notas de um aluno e calcule a média.
5. Com base na média, crie um fator com as categorias "aprovado" ou "reprovado".
6. Monte uma lista com nome, idade e vetor de notas de um aluno fictício.
7. Crie um data frame com 3 alunos e 3 colunas: nome, idade e situação ("aprovado"/"reprovado").

Parte 2 – Exercícios Complementares

8. Crie uma **matriz** com 3 linhas e 3 colunas representando notas de 3 alunos em 3 provas.
   - Dê nomes às linhas (nomes dos alunos) e colunas (P1, P2, P3).
   - Calcule a média por aluno (linha) e por prova (coluna).
9. Construa uma **lista** que armazene os dados de cada aluno:
   - Nome, idade, notas (vetor), média, situação.
   - Acesse os elementos usando `$` e índices.
10. Crie dois **data frames**:
    - Um com dados cadastrais (`nome`, `idade`)
    - Outro com dados acadêmicos (`nome`, `média`, `situação`)
    - Use a função `merge()` para juntar os dois com base no nome do aluno (experimente os tipos `inner`, `left`, `full`).
11. Use operadores lógicos para:
    - Filtrar alunos com média acima de 7.
    - Verificar quem tem mais de 20 anos e foi aprovado.
    - Identificar os reprovados com menos de 18 anos.
12. Crie um vetor com os nomes dos meses do semestre e use-o como fator ordenado.
    - Associe esse fator aos vetores de despesas e receitas criados antes.
    - Construa um `data.frame` com `mês`, `receita`, `despesa`, `lucro`.
13. Exporte esse `data.frame` para um arquivo `.csv` usando `write.csv()`.

14. Apresente 5 novos comandos (operações) em matrizes.
15. Apresente 5 novos comandos (operações) em listas.
16. Apresente 5 novos comandos (operações) em dataframes.

Resumo

- Tipos básicos: numérico, caractere, lógico
- Vetores e suas operações
- Fatores para variáveis categóricas
- Listas com múltiplos tipos
- Matrizes bidimensionais
- Introdução aos data frames