Café com estatística e R

Treinamento 1 - Tipos de variáveis, escalas e uma introdução ao R

Marcelo Teixeira Paiva

2025-09-19

Abstract

Relatório do primeiro treinamento onde foi apresentado uma introdução ao R e os conceitos de tipos de variáveis e escalas.

Índice

Lista de Figuras

Lista de Tabelas

Chapter 1

Fundamentos de R para análise de dados

1.1 Conhecimentos básicos de R

1.1.1 Instalação do R e RStudio

R é uma linguagem para computação estatística, enquanto **RStudio** é um ambiente de desenvolvimento integrado (IDE) que facilita o trabalho. Ou seja, o R é quem faz o trabalho pesado e o RStudio é uma das várias maneiras de se usar o R com menos esforço.

Processo de instalação: 1. Baixe o R em: https://cran.r-project.org/ 2. Baixe o RStudio em: https://posit.co/products/open-source/rstudio/?sid=1 3. Siga os passos de instalação de cada um deles em suas próprias páginas.

1.1.2 Sintaxe Básica

Imagine que você vai ler um artigo. Você imprime esse documento e inicia sua leitura, mas começa sentir sono e resolve parar e ir tomar um café. Quando você retorna para continuar sua leitura, seu artigo sumiu! Pior que isso, você também esqueceu onde havia parado de ler!

Então você precisa novamente imprimir o documento e iniciar sua leitura novamente. Agora imagine que isso acontece a cada vez que para de ler e se distancia do seu documento. Seria um sofrimento ler qualquer artigo, uma vez que sempre seria necessário imprimir e ler o documento de uma vez.

O mesmo ocorre na análise de dados e computação em geral, nós queremos ter uma forma de ler ou registrar um dado e depois poder retornar a usá-lo sem grandes problemas. Para isso usamos variáveis (que não é a mesma variável da estatística). Então, no R, uma variável representa um nome associado a um dado gravado na memória. Por ser somente um nome, não há restrições para o que ele nomeia (o tipo de dado), somente não se aceita que ele seja um nome feio, que usa caractéres proibidos (numeros no início, “$”“,”“, “,”).

```
# | label: creating_variables

# para criar comentários comece a linha com #
# esses comentários são desconsiderados pelo interpretador do R (não processados)
```

```r
# forma de atribuição mais comum
x <- 10
# forma menos comum
y = 20
# Atribuição reversa - para aqueles que vivem no Upside Down
30 -> z

# R é case-sensitive (diferencia maiusculas de minusculas)
Var1 <- 5
var1 <- 10

# Boas práticas para nomear variável
# Nunca:
# - Começar com números: 2var (incorreto)
# - Usar espaços ou carácteres especiais: minha variavel (incorreto)
# - Usar palavras reservadas: mean, if, for

# Use nomes descritivos, quem lê seu código não sabe o que você pensou
p_valor_teste_t <- 0.032
ic_95_inferior <- 12.3
ic_95_superior <- 18.7
```

1.1.3 Tipos de Dados

Numeric (double/integer):

```r
# | label: data_types_numeric

# Números reais (padrão)
altura <- 1.75
peso <- 68.5
# "numeric"
class(altura)
```

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

```r
class(1)
```

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

```r
# Inteiros (com L)
n_amostras <- 100L
# "integer"
class(n_amostras)
```

```
[1] "integer"
```

Character (texto):

```r
# | label: data_types_character

tratamento <- "vacina"
```

```r
# c() é um vetor (agrupamento de dados atômicos)
grupo <- c("controle", "tratado", "placebo")
# "character"
class(tratamento)
```

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

Logical (booleano, verdadeiro/falso):

```r
# | label: data_types_logical

significativo <- TRUE
hipotese_nula <- FALSE
p_valor <- 0.01
# operações lógicas: Retorna TRUE ou FALSE
p_valor < 0.05
```

```
[1] TRUE
```

```r
class(hipotese_nula)
```

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

1.1.4 Vetores e Operações Básicas

Os vetores são a estrutura fundamental do R. **Tudo é vetor em sua essência**!

```r
# | label: data_types_vector

# Como criar vetores
dados <- c(23, 45, 12, 67, 34)
sequencia <- 1:10
seq_regular <- seq(0, 1, by=0.1)
repeticao <- rep(c(0,1), times=5)

# vetor nomeado
idades <- c(fulano=21, cicrano=43)
names(idades)
```

```
[1] "fulano"  "cicrano"
```

```r
# Operações vetorizadas são realizadas elemento por elemento
x <- c( 1,  2,  3,  4,  5)
y <- c(10, 20, 30, 40, 50)

x + y
```

```
[1] 11 22 33 44 55
```

```r
x * 2
```

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

```r
x^2
```

```
[1]  1  4  9 16 25
```

```r
sqrt(x)
```

```
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

```r
# Operações em vetores de tamanho diferente
# Cuidado! porque ocorre reciclagem do menor vetor
c(1, 2, 3) + c(10, 20)
```

```
Warning in c(1, 2, 3) + c(10, 20): comprimento do objeto maior não é múltiplo
do comprimento do objeto menor
```

```
[1] 11 22 13
```

1.1.5 Funções Básicas

Funções possuem um padrão nome_da_funcao(argumento1, argumento2, ...). Ela é um bloco de código com uma finalidade específica, que abstrai a complexidade de como é feito algo para quem a usa. Então, por exemplo, se uso uma função media(x), eu não preciso saber o "como" e somente o que ela faz (calcula a média de um grupo de elementos em x).

Funções também são úteis quando repetimos um bloco de código em vários momentos de uma análise, pois, podemos definir uma função para executar esse bloco de código uma vez e depois só executá-la (princípio DRY).

```r
# | label: functions_basic

dados <- c(23, 45, 12, 67, 34, 28, 51)

# funções do dia-a-dia
sum(dados)        # Soma
```

```
[1] 260
```

```r
mean(dados)       # Média aritmética
```

```
[1] 37.14286
```

```r
median(dados)     # Mediana
```

```
[1] 34
```

```r
var(dados)        # Variância amostral (n-1)
```

```
[1] 345.1429
```

```r
sd(dados)         # Desvio padrão
```

```
[1] 18.57802
```

```r
min(dados)        # Mínimo
```

```
[1] 12
```

```r
max(dados)        # Máximo
```

```
[1] 67
```

```r
range(dados)      # Min e Max
```

```
[1] 12 67
```

```
quantile(dados)   # Quartis
```

```
  0%  25%  50%  75% 100%
12.0 25.5 34.0 48.0 67.0
```

```
summary(dados)    # Resumo estatístico
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  12.00   25.50   34.00   37.14   48.00   67.00
```

```
# Outras funções úteis
# Tamanho do vetor
length(dados)
```

```
[1] 7
```

```
# Ordenação
sort(dados, decreasing = FALSE)
```

```
[1] 12 23 28 34 45 51 67
```

```
# Valores únicos
unique(dados)
```

```
[1] 23 45 12 67 34 28 51
```

```
# Tabela de frequências
dados <- c(rep("a", 2), rep("b", 4), rep("c", 8), rep("d", 1))
table(dados)
```

```
dados
a b c d
2 4 8 1
```

```
prop.table(table(dados))
```

```
dados
         a          b          c          d
0.13333333 0.26666667 0.53333333 0.06666667
```

1.1.6 Operadores Aritméticos e Lógicos

Operadores Aritméticos:

```
# | label: arithmetics_operations

# Básicos
10 + 5    # Adição
```

```
[1] 15
```

```
10 - 5    # Subtração
```

```
[1] 5
```

```r
10 * 5    # Multiplicação
```

```
[1] 50
```

```r
10 / 5    # Divisão
```

```
[1] 2
```

```r
10 ^ 2    # Potenciação
```

```
[1] 100
```

```r
10 ** 2   # Potenciação
```

```
[1] 100
```

```r
10 %% 3   # Módulo (resto): 1
```

```
[1] 1
```

```r
10 %/% 3  # Divisão inteira: 3
```

```
[1] 3
```

```r
amostra <- sample(0:200, 1e6, replace = TRUE)
media <- sum(amostra) / length(amostra)
variancia <- sum((amostra - mean(amostra))^2) / (length(amostra) - 1)
```

Operadores Lógicos:

```r
# | label: boolean_operations

# Comparação
5 > 3      # maior
```

```
[1] TRUE
```

```r
5 < 3      # menor
```

```
[1] FALSE
```

```r
5 >= 3     # maior ou igual
```

```
[1] TRUE
```

```r
5 <= 3     # menor ou igual
```

```
[1] FALSE
```

```r
5 == 3     # igual
```

```
[1] FALSE
```

```r
5 != 3     # diferente
```

```
[1] TRUE
```

```r
"a" == "b"
```

```
[1] FALSE
```

```r
# Operadores booleanos
p <- TRUE
q <- FALSE
# operação   conectivo
!p           # NEGAÇÃO
```

```
[1] FALSE
```

```r
p & q        # E - conjunção (só é V em VV)
```

```
[1] FALSE
```

```r
p | q        # OU - disjunção inclusiva (só é F em FF)
```

```
[1] TRUE
```

```r
xor(p, q)    # OU OU - disjunção exclusiva (é F sempre que iguais - VV, FF)
```

```
[1] TRUE
```

```r
!p | q       # equivalente à condicional
```

```
[1] FALSE
```

```r
(!p | q) & (!q | p) # equivalente à bicondicional
```

```
[1] FALSE
```

```r
10 < 12 & 12 > 5
```

```
[1] TRUE
```

```r
# short circuit evaluation
10 > 12 && nao_existo
```

```
[1] FALSE
```

```r
10 < 12 || nao_existo
```

```
[1] TRUE
```

```r
# Vetorização
idades <- c(18, 25, 30, 17, 22)
idades >= 18
```

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

1.2 Estruturas de dados

1.2.1 Matrizes

Estruturas bidimensionais com elementos do **mesmo tipo**.

```r
# | label: matrices

# Criação de matrizes
matriz1 <- matrix(1:12, nrow=3, ncol=4)
matriz2 <- matrix(1:12, nrow=3, ncol=4, byrow=TRUE)
```

```r
# Matriz de correlação
dados <- matrix(rnorm(100), ncol=5)
cor_matrix <- cor(dados)

# Operações matriciais
A <- matrix(c(1,2,3,4), nrow=2)
B <- matrix(c(5,6,7,8), nrow=2)

A + B          # Soma elemento por elemento
```

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

```r
A * B          # Multiplicação elemento por elemento
```

```
     [,1] [,2]
[1,]    5   21
[2,]   12   32
```

```r
A %*% B        # Multiplicação matricial verdadeira
```

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

```r
t(A)           # Transposta
```

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

```r
solve(A)       # Inversa (se existir)
```

```
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

```r
det(A)         # Determinante
```

```
[1] -2
```

```r
# Dimensões
dim(A)
```

```
[1] 2 2
```

```r
nrow(A)
```

```
[1] 2
```

```r
ncol(A)
```

```
[1] 2
```

1.2.2 Data Frames

São basicamente tabelas com colunas de **variados tipos** em que cada linha representa um registro (planilha do excel). É o principal tipo de dado com que trabalhamos na prática. Todas as colunas no `data.frame` devem apresentar o mesmo tamanho.

```
# | label: data_frames

# Criação
df <- data.frame(
  id = 1:5,
  tratamento = c("A", "B", "A", "B", "A"),
  peso_inicial = c(65.2, 70.1, 68.5, 72.3, 66.8),
  peso_final = c(68.1, 71.5, 71.2, 73.8, 69.5),
  melhorou = c(TRUE, TRUE, TRUE, TRUE, NA)
)

# Estrutura e resumo
str(df)         # Estrutura do data.frame
```

```
'data.frame':   5 obs. of  5 variables:
 $ id          : int  1 2 3 4 5
 $ tratamento  : chr  "A" "B" "A" "B" ...
 $ peso_inicial: num  65.2 70.1 68.5 72.3 66.8
 $ peso_final  : num  68.1 71.5 71.2 73.8 69.5
 $ melhorou    : logi  TRUE TRUE TRUE TRUE NA
```

```
summary(df)     # Resumo estatístico simples de cada coluna
```

```
       id     tratamento         peso_inicial     peso_final    melhorou
 Min.   :1   Length:5           Min.   :65.20   Min.   :68.10   Mode:logical
 1st Qu.:2   Class :character   1st Qu.:66.80   1st Qu.:69.50   TRUE:4
 Median :3   Mode  :character   Median :68.50   Median :71.20   NA's:1
 Mean   :3                      Mean   :68.58   Mean   :70.82
 3rd Qu.:4                      3rd Qu.:70.10   3rd Qu.:71.50
 Max.   :5                      Max.   :72.30   Max.   :73.80
```

```
head(df)        # Primeiras x linhas
```

```
  id tratamento peso_inicial peso_final melhorou
1  1          A         65.2       68.1     TRUE
2  2          B         70.1       71.5     TRUE
3  3          A         68.5       71.2     TRUE
4  4          B         72.3       73.8     TRUE
5  5          A         66.8       69.5       NA
```

```
tail(df)        # Últimas x linhas
```

```
  id tratamento peso_inicial peso_final melhorou
1  1          A         65.2       68.1     TRUE
2  2          B         70.1       71.5     TRUE
3  3          A         68.5       71.2     TRUE
4  4          B         72.3       73.8     TRUE
5  5          A         66.8       69.5       NA
```

```r
# Acessando colunas
df$peso_inicial
```

```
[1] 65.2 70.1 68.5 72.3 66.8
```

```r
df[["peso_inicial"]]
```

```
[1] 65.2 70.1 68.5 72.3 66.8
```

```r
df[, "peso_inicial"]
```

```
[1] 65.2 70.1 68.5 72.3 66.8
```

```r
df[, 3]
```

```
[1] 65.2 70.1 68.5 72.3 66.8
```

```r
df[c(1, 3), 3]
```

```
[1] 65.2 68.5
```

```r
# Criando nova variável no data.frame
df$ganho_peso <- df$peso_final - df$peso_inicial
df$ganho_percentual <- (df$ganho_peso / df$peso_inicial) * 100

head(df)
```

```
  id tratamento peso_inicial peso_final melhorou ganho_peso ganho_percentual
1  1          A         65.2       68.1     TRUE        2.9         4.447853
2  2          B         70.1       71.5     TRUE        1.4         1.997147
3  3          A         68.5       71.2     TRUE        2.7         3.941606
4  4          B         72.3       73.8     TRUE        1.5         2.074689
5  5          A         66.8       69.5       NA        2.7         4.041916
```

1.2.3 Listas

Estruturas mais flexíveis - podem conter elementos de **tipos e tamanhos diferentes**.

```r
# | label: lists

# Lista com resultados de uma análise estatística
resultado_teste <- list(
  nome_teste = "Teste t de Student",
  estatistica_t = 2.453,
  graus_liberdade = 48,
  p_valor = 0.018,
  intervalo_confianca = c(1.23, 5.67),
  dados_originais = df,
  matriz_cov = matrix(rnorm(9), 3, 3)
)

# Acessando elementos
resultado_teste$p_valor
```

```
[1] 0.018
```

```r
resultado_teste[["p_valor"]]
```

```
[1] 0.018
```

```r
resultado_teste[[4]]
```

```
[1] 0.018
```

```r
teste_t <- t.test(df$peso_final, df$peso_inicial, paired=TRUE)
str(teste_t)
```

```
List of 10
 $ statistic  : Named num 6.89
  ..- attr(*, "names")= chr "t"
 $ parameter  : Named num 4
  ..- attr(*, "names")= chr "df"
 $ p.value    : num 0.00232
 $ conf.int   : num [1:2] 1.34 3.14
  ..- attr(*, "conf.level")= num 0.95
 $ estimate   : Named num 2.24
  ..- attr(*, "names")= chr "mean difference"
 $ null.value : Named num 0
  ..- attr(*, "names")= chr "mean difference"
 $ stderr     : num 0.325
 $ alternative: chr "two.sided"
 $ method     : chr "Paired t-test"
 $ data.name  : chr "df$peso_final and df$peso_inicial"
 - attr(*, "class")= chr "htest"
```

```r
names(teste_t)
```

```
 [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
 [6] "null.value"  "stderr"      "alternative" "method"      "data.name"
```

1.2.4 Fatores

Variáveis categóricas com níveis fixos - essencial para modelos estatísticos.

```r
# | label: factors

# Criação de fatores
sexo <- factor(c("M", "F", "F", "M", "F"), levels = c("M", "F"))
levels(sexo)
```

```
[1] "M" "F"
```

```r
# Recodificação
levels(sexo) <- c("Masculino", "Feminino")

# Fator ordenado
educacao <- factor(
  c("Médio", "Superior", "Fundamental", "Superior", "Médio"),
  levels = c("Fundamental", "Médio", "Superior"),
  ordered = TRUE
```

```
)
as.integer((educacao))
```

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

```
# GLMs e ANOVA tratam fatores como dummies ou como variáveis discretas
df$tratamento[5] <- "C"
df$grupo <- df$tratamento
lm(peso_final ~ grupo, data=df)
```

```

Call:
lm(formula = peso_final ~ grupo, data = df)

Coefficients:
(Intercept)       grupoB       grupoC
      69.65         3.00        -0.15
```

```
df$grupo <- factor(df$tratamento)
lm(peso_final ~ grupo, data=df)
```

```

Call:
lm(formula = peso_final ~ grupo, data = df)

Coefficients:
(Intercept)       grupoB       grupoC
      69.65         3.00        -0.15
```

```
df$grupo <- factor(df$tratamento, ordered = TRUE)
lm(peso_final ~ grupo, data=df)
```

```

Call:
lm(formula = peso_final ~ grupo, data = df)

Coefficients:
(Intercept)      grupo.L      grupo.Q
    70.6000      -0.1061      -2.5107
```

1.2.5 Arrays

Generalizações de matrizes para **múltiplas dimensões**.

```
# | label: arrays

# Array de 3 dimensões (exemplo: medidas x indivíduos x tempo)
n_pacientes <- 5
n_tempos <- 3
peso_inicial <- rnorm(n_pacientes, mean = 70, sd = 5)
altura_inicial <- rnorm(n_pacientes, mean = 170, sd = 10)
idade_inicial <- rpois(n_pacientes, lambda = 23)
```

```r
# matriz vazia
peso_tempo <- matrix(nrow = n_pacientes, ncol = n_tempos)
altura_tempo <- matrix(nrow = n_pacientes, ncol = n_tempos)
idade_tempo <- matrix(nrow = n_pacientes, ncol = n_tempos)

for(i in 1:n_pacientes) {
  peso_tempo[i, ] <- peso_inicial[i] + cumsum(c(0, rnorm(n_tempos-1, mean=0.5, sd=1)))
  altura_tempo[i, ] <- altura_inicial[i] + rnorm(n_tempos, mean=0, sd=0.5)
  idade_tempo[i, ] <- idade_inicial[i] + c(0, 0.25, 0.5)
}

imc_tempo <- peso_tempo / (altura_tempo/100)^2

medidas_tempo <- array(
  c(t(peso_tempo), t(altura_tempo), t(imc_tempo), t(idade_tempo)),
  dim = c(n_pacientes, n_tempos, 4),
  dimnames = list(
    paste("Paciente", 1:5),
    c("Mês_0", "Mês_3", "Mês_6"),
    c("Peso", "Altura", "IMC", "Idade")
  )
)

medidas_tempo[1, , ] # paciente 1
```

```
          Peso   Altura      IMC Idade
Mês_0 68.06854 165.3407 24.89928 24.00
Mês_3 71.63160 160.9087 27.66594 10.50
Mês_6 71.43953 161.1359 27.51402 28.25
```

```r
medidas_tempo[, , 4] # idade
```

```
           Mês_0 Mês_3 Mês_6
Paciente 1 24.00 10.50 28.25
Paciente 2 24.25 19.00 28.50
Paciente 3 24.50 19.25 22.00
Paciente 4 10.00 19.50 22.25
Paciente 5 10.25 28.00 22.50
```

```r
medidas_tempo[, 2, ] # mes 2
```

```
              Peso   Altura      IMC Idade
Paciente 1 71.63160 160.9087 27.66594 10.50
Paciente 2 73.99008 186.0141 21.38364 19.00
Paciente 3 73.34040 184.5639 21.53030 19.25
Paciente 4 73.86377 185.8718 21.37984 19.50
Paciente 5 71.60735 161.8853 27.32390 28.00
```

```r
n <- 1000
dados_epi <- data.frame(
  Sexo = sample(c("M", "F"), n, replace = TRUE),
  Idade = sample(c("0-20", "21-40", "41-60", "60+"), n, replace = TRUE),
```

```
  Exposicao = sample(c("Sim", "Não"), n, replace = TRUE, prob = c(0.3, 0.7)),
  Doenca = sample(c("Presente", "Ausente"), n, replace = TRUE, prob = c(0.1, 0.9))
)
tabela_4d <- table(dados_epi)
print(dim(tabela_4d))  # 2 x 4 x 2 x 2
```

```
[1] 2 4 2 2
```

```
# Análise de odds ratio estratificado
for(sexo in c("M", "F")) {
  for(idade in unique(dados_epi$Idade)) {
    subtabela <- tabela_4d[sexo, idade, , ]
    if(all(subtabela > 0)) {
      #|                | desfecho: Sim  | desfecho: Não |
      #|----------------|----------------|---------------|
      #| preditor: Sim  | A              | B             |
      #| preditor: Não  | C              | D             |
      # OR = a*d/b*c

      OR <- (subtabela[2,2] * subtabela[1,1]) /
            (subtabela[2,1] * subtabela[1,2])
      cat(sprintf("OR para %s, %s: %.2f\n", sexo, idade, OR))
    }
  }
}
```

```
OR para M, 21-40: 1.16
OR para M, 0-20: 0.87
OR para M, 60+: 1.50
OR para M, 41-60: 0.42
OR para F, 21-40: 0.75
OR para F, 0-20: 1.83
OR para F, 60+: 2.14
OR para F, 41-60: 1.06
```

1.2.6 Indexação e Seleção de Dados

```
# | label: indexing

# VETORES
x <- c(10, 20, 30, 40, 50)
x[2]
```

```
[1] 20
```

```
x[c(1,3,5)]
```

```
[1] 10 30 50
```

```
x[-2]
```

```
[1] 10 30 40 50
```

```r
x[x > 25]
```

```
[1] 30 40 50
```

```r
# MATRIZES [linha, coluna]
mat <- matrix(1:12, nrow=3)
mat[2, 3]
```

```
[1] 8
```

```r
mat[2, ]
```

```
[1]  2  5  8 11
```

```r
mat[, 3]
```

```
[1] 7 8 9
```

```r
mat[1:2, 3:4]
```

```
     [,1] [,2]
[1,]    7   10
[2,]    8   11
```

```r
# DATA FRAMES
df[2, 3]
```

```
[1] 70.1
```

```r
df[2, ]
```

```
  id tratamento peso_inicial peso_final melhorou ganho_peso ganho_percentual
2  2          B         70.1       71.5     TRUE        1.4         1.997147
  grupo
2     B
```

```r
df[, "peso_inicial"]
```

```
[1] 65.2 70.1 68.5 72.3 66.8
```

```r
df$peso_inicial
```

```
[1] 65.2 70.1 68.5 72.3 66.8
```

```r
df[df$tratamento == "A", ]
```

```
  id tratamento peso_inicial peso_final melhorou ganho_peso ganho_percentual
1  1          A         65.2       68.1     TRUE        2.9         4.447853
3  3          A         68.5       71.2     TRUE        2.7         3.941606
  grupo
1     A
3     A
```

```r
df[df$ganho_peso > 2 & df$tratamento == "A", c("id", "ganho_peso")]
```

```
  id ganho_peso
1  1        2.9
3  3        2.7
```

```r
# LISTAS
lista <- list(a=1:5, b=matrix(1:4,2), c="texto")
lista[[1]]
```

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

```r
lista$a
```

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

```r
lista[["b"]]
```

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

Chapter 2

Tipos de variáveis e escalas de mensuração e precisão