

# Dia 1: Fundamentos e Ambiente R

Vinícius Silva Junqueira

2025-10-07

## Contents

<b>OBJETIVO: Objetivos do Dia 1</b>	<b>2</b>
<b>MATERIAL: Parte 1: Ambientação e Setup (19h00 - 20h30)</b>	<b>3</b>
1.1 Por que R, GitHub e IA? . . . . .	3
Por que R? . . . . .	3
Por que GitHub? . . . . .	3
Por que IA (ChatGPT e Claude)? . . . . .	3
1.2 Verificação das Instalações . . . . .	3
Verificar R . . . . .	3
Verificar RStudio . . . . .	4
Verificar Git . . . . .	4
1.3 Instalação de Pacotes Essenciais . . . . .	4
Carregar pacotes . . . . .	4
1.4 Configuração do GitHub . . . . .	4
Criar Conta no GitHub . . . . .	4
Configurar Git Local . . . . .	4
Criar Personal Access Token (PAT) . . . . .	5
Salvar Token Localmente . . . . .	5
1.5 Clonando o Repositório do Curso . . . . .	5
Via RStudio (Recomendado) . . . . .	5
Via Terminal . . . . .	5
1.6 Estrutura do Projeto . . . . .	5
Por que usar Projetos RStudio? . . . . .	6
1.7 O Pacote <code>here</code> . . . . .	6
<b>INTERVALO: INTERVALO (20h30 - 20h50)</b>	<b>7</b>
<b>GRAFICO: Parte 2: Fundamentos do R (20h50 - 22h00)</b>	<b>7</b>
2.1 R como Calculadora . . . . .	7
2.2 Objetos e Atribuição . . . . .	7
Regras para Nomes de Objetos . . . . .	8
2.3 Tipos de Dados Fundamentais . . . . .	8
Numeric (N meros) . . . . .	8
Character (Texto/String) . . . . .	9
Logical (L gico) . . . . .	9
Factor (Fator/Categ rico) . . . . .	9
Convers o entre Tipos . . . . .	10
2.4 Vetores . . . . .	10
Criando Vetores . . . . .	10
Propriedades de Vetores . . . . .	11

Operações Vetorizadas . . . . .	12
2.5 Indexação de Vetores . . . . .	12
Por Posição . . . . .	12
Por Condição Lógica . . . . .	13
Por Nome . . . . .	13
2.6 Valores Ausentes (NA) . . . . .	14
2.7 Listas . . . . .	15
2.8 Data Frames . . . . .	15
Acessar Colunas . . . . .	17
Acessar Linhas . . . . .	17
Acessar Células Específicas . . . . .	18
Adicionar Colunas . . . . .	18
2.9 Funções de Exploração . . . . .	18
<b>EXERCÍCIO: Exercícios Práticos</b>	<b>20</b>
Exercício 1: Calculadora e Objetos . . . . .	20
Exercício 2: Vetores . . . . .	20
Exercício 3: Data Frames . . . . .	20
Exercício 4: Análise Real . . . . .	20
<b>OBJETIVO: Prática Guiada: Primeiro Script</b>	<b>21</b>
Passo 1: Criar Script . . . . .	21
Passo 2: Escrever Código . . . . .	21
Passo 3: Executar . . . . .	22
<b>GITHUB: Primeiro Commit no GitHub</b>	<b>22</b>
Via RStudio (Recomendado) . . . . .	22
Via Terminal . . . . .	22
Verificar . . . . .	23
<b>MATERIAL: Para Casa</b>	<b>23</b>
<b>Recursos Adicionais</b>	<b>23</b>
Documentação . . . . .	23
Prática . . . . .	23
Comunidades . . . . .	23
<b>Dúvidas Frequentes</b>	<b>23</b>
<b>CURSO: Conclusão do Dia 1</b>	<b>24</b>

---

## OBJETIVO: Objetivos do Dia 1

Ao final desta aula, você será capaz de:

- [OK] Configurar completamente seu ambiente de trabalho (R, RStudio, Git, GitHub)
  - [OK] Entender os tipos de dados e estruturas fundamentais do R
  - [OK] Criar e manipular vetores, listas e data frames
  - [OK] Realizar operações básicas de indexação
  - [OK] Organizar projetos com RStudio Projects
  - [OK] Fazer seu primeiro commit no GitHub
  - [OK] Usar o pacote `here` para caminhos relativos
-

# MATERIAL: Parte 1: Ambientação e Setup (19h00 - 20h30)

## 1.1 Por que R, GitHub e IA?

Por que R?

R é uma linguagem de programação estatística amplamente utilizada em:

- **Pesquisa científica:** análise de dados experimentais
- **Saúde pública:** epidemiologia, ensaios clínicos
- **GRAFICO: Economia e finanças:** modelagem econométrica, séries temporais
- **Ecologia:** análise de biodiversidade, modelos populacionais
- **Marketing e vendas:** segmentação, análise de campanhas
- **Ciências sociais:** análise de surveys, dados censitários

**Vantagens:** - **Gratuito e open-source** - **Comunidade ativa** e acolhedora (R-Ladies, Stack Overflow) - **Pacotes especializados** para praticamente qualquer análise - **Reprodutibilidade:** RMarkdown permite documentar análises - **Visualizações de alta qualidade:** ggplot2 é referência mundial

Por que GitHub?

GitHub é uma plataforma de versionamento de código que permite:

- **NOTA: Histórico completo** de todas as mudanças
- **WORKFLOW: Colaboração** eficiente em equipe
- **Backup automático** na nuvem
- **Portfólio público** de projetos
- **Compartilhamento** fácil de análises

Por que IA (ChatGPT e Claude)?

Ferramentas de IA generativa aceleram o aprendizado:

- **DICA: Explicações personalizadas** de conceitos difíceis
- **Debugging assistido:** entender erros rapidamente
- **NOTA: Documentação automática:** comentar código
- **INICIO: Produtividade:** sugestões de código e otimizações
- **CURSO: Tutor 24/7:** responde dúvidas a qualquer momento

**ATENCAO: Atenção:** IA é uma ferramenta auxiliar. Sempre entenda o código antes de usar!

---

## 1.2 Verificação das Instalações

Antes de começar, vamos verificar se tudo está instalado corretamente.

Verificar R

```
# Versão do R (deve ser 4.0 ou superior)
R.version.string
```

```
#> [1] "R version 4.5.1 (2025-06-13)"
```

```
# Idealmente 4.3.0 ou mais recente
```

## Verificar RStudio

No menu: **Help About RStudio**

Versão recomendada: 2023.09 ou superior

## Verificar Git

No **Terminal** do RStudio (aba ao lado de Console):

```
git --version
```

Deve retornar algo como: `git version 2.40.0`

Se não funcionar: - **Windows:** Instale Git Bash (<https://git-scm.com/>) - **Mac:** Instale Xcode Command Line Tools (`xcode-select --install`) - **Linux:** `sudo apt-get install git` (Ubuntu/Debian)

---

## 1.3 Instalação de Pacotes Essenciais

Vamos instalar os pacotes que usaremos durante o curso:

```
# Instalar pacotes (execute apenas uma vez)
install.packages(c(
  "tidyverse", # Conjunto de pacotes para ciência de dados
  "here",      # Caminhos relativos seguros
  "janitor",   # Limpeza de dados
  "skimr",     # Resumos estatísticos
  "readxl",    # Leitura de arquivos Excel
  "writexl",   # Escrita de arquivos Excel
  "rmarkdown", # Relatórios reproduzíveis
  "usethis",   # Ferramentas de desenvolvimento
  "gitcreds"   # Gerenciamento de credenciais Git
))
```

### Carregar pacotes

```
# Carregar pacotes (execute sempre que iniciar uma sess o)
library(tidyverse) # Carrega dplyr, ggplot2, tidyr, etc.
library(here)      # Para caminhos de arquivos
```

DICA: Dica: `install.packages()` uma vez, `library()` sempre!

---

## 1.4 Configuração do GitHub

### Criar Conta no GitHub

Se ainda não tem: <https://github.com/signup>

**Dicas:** - Use um email profissional/acadêmico - Username curto e profissional (seu nome ou variação) - Ative autenticação de dois fatores (2FA)

### Configurar Git Local

```
# Configure seu nome e email (use os mesmos do GitHub)
usethis::use_git_config(
```

```
user.name = "Seu Nome Completo",
user.email = "seu-email@example.com"
)
```

Ou no Terminal:

```
git config --global user.name "Seu Nome Completo"
git config --global user.email "seu-email@example.com"
```

### Criar Personal Access Token (PAT)

O PAT é como uma senha especial para o Git:

```
# Abre o GitHub para criar token
usethis::create_github_token()
```

**No GitHub:** 1. Dê um nome descritivo (ex: “Curso R 2025”) 2. Selecione expiração (90 dias ou mais) 3. Marque apenas: **repo**, **workflow**, **gist** 4. Clique em “Generate token” 5. **COPIE O TOKEN** (não conseguiremos ver novamente!)

### Salvar Token Localmente

```
# Cole o token quando solicitado
gitcreds::gitcreds_set()
```

Pronto! Agora você pode usar Git/GitHub sem digitar senha toda hora.

---

## 1.5 Clonando o Repositório do Curso

### Via RStudio (Recomendado)

1. **File New Project Version Control Git**
2. Cole a URL: <https://github.com/seu-usuario/curso-r-github-ia.git>
3. Escolha onde salvar no seu computador
4. Marque “Open in new session”
5. **Create Project**

### Via Terminal

```
# Navegue até onde quer salvar
cd ~/Documents

# Clone o repositório
git clone https://github.com/seu-usuario/curso-r-github-ia.git

# Entre na pasta
cd curso-r-github-ia

# Abra o projeto no RStudio (no Windows, apenas dê um duplo clique no .Rproj)
```

---

## 1.6 Estrutura do Projeto

Um projeto bem organizado tem esta estrutura:

```

curso-r-github-ia/
  curso-r-github-ia.Rproj # Arquivo do projeto (abra sempre por aqui!)
  README.md              # Descrição do projeto
  .gitignore              # Arquivos que o Git deve ignorar

  data/                  # Dados
    raw/                 # Dados originais (NUNCA modificar!)
    processed/           # Dados limpos/processados

  scripts/               # Scripts R
    01_import.R
    02_clean.R
    03_analyze.R

  output/                # Resultados
    figures/             # Gráficos salvos
    tables/              # Tabelas exportadas

  materiais/             # Material do curso
    dial_fundamentos.Rmd
    ...

  docs/                  # Relatórios finais

```

### Por que usar Projetos RStudio?

- [OK] **Working directory automático:** sempre aponta para a pasta do projeto
- [OK] **Portabilidade:** funciona em qualquer computador
- [OK] **Organização:** mantém tudo junto
- [OK] **Integração com Git:** painel Git aparece automaticamente

## 1.7 O Pacote here

O `here` resolve problemas de caminhos de arquivos:

```

# Caminho absoluto (RUIM - não funciona em outro computador)
# dados <- read_csv("C:/Users/Vinicius/Documents/curso/data/dados.csv")

```

```

# Caminho relativo com here (BOM - funciona em qualquer lugar)
here::here() # Mostra a raiz do projeto

```

```

#> [1] "/Users/viniciusjunqueira/Library/CloudStorage/OneDrive-Pessoal/Cursos/curso-r-github-ia"

# Como usar
caminho_dados <- here("data", "raw", "exemplo.csv")
print(caminho_dados)

```

```

#> [1] "/Users/viniciusjunqueira/Library/CloudStorage/OneDrive-Pessoal/Cursos/curso-r-github-ia/data/raw"

# Na prática:
# dados <- read_csv(here("data", "raw", "dados.csv"))

```

Sempre use `here()` para referenciar arquivos!

## INTERVALO: INTERVALO (20h30 - 20h50)

Aproveite para: - Tomar água/café - Conferir se todas as instalações funcionaram - Tirar dúvidas no grupo

---

## GRAFICO: Parte 2: Fundamentos do R (20h50 - 22h00)

### 2.1 R como Calculadora

O jeito mais simples de começar: usar R como calculadora avançada!

```
# Operações básicas
2 + 3          # Adição

#> [1] 5

10 - 4         # Subtração

#> [1] 6

5 * 6          # Multiplicação

#> [1] 30

20 / 4         # Divisão

#> [1] 5

2^3           # Potência

#> [1] 8

sqrt(16)       # Raiz quadrada

#> [1] 4

log(10)        # Logaritmo natural

#> [1] 2.302585

log10(100)     # Logaritmo base 10

#> [1] 2

# Ordem de operações (PEMDAS)
2 + 3 * 4      # 14, não 20!

#> [1] 14

(2 + 3) * 4    # 20, com parênteses

#> [1] 20
```

---

### 2.2 Objetos e Atribuição

No R, guardamos valores em **objetos** usando `<-` (atalho: `Alt + -`)

```
# Criar objetos
x <- 5
y <- 10
```

```

nome <- "Vinicius"
aprovado <- TRUE

# Ver conteúdo
x

#> [1] 5

print(y)

#> [1] 10

# Usar em operações
resultado <- x + y
resultado

#> [1] 15

# Sobrescrever
x <- 20
x # Agora vale 20, não mais 5!

#> [1] 20

```

## Regras para Nomes de Objetos

[OK] Permitido:

```

idade <- 30
idade_media <- 25
idadeMedia <- 25 # camelCase (menos comum em R)
idade2 <- 27

```

[X] NÃO permitido:

```

2idade <- 30 # Não pode começar com número
idade-media <- 25 # Hífen não é permitido (use _)
minha idade <- 30 # Sem espaços!

```

DICA: **Convenção R:** Use `snake_case` (palavras separadas por `_`)

---

## 2.3 Tipos de Dados Fundamentais

O R tem 6 tipos básicos, mas vamos focar nos 4 principais:

### Numeric (Números)

```

# Inteiros e decimais
altura <- 1.75
peso <- 70
idade <- 30

# Verificar tipo
class(altura)

#> [1] "numeric"

```



```
typeof(altura) # double = n mero com decimais
```

```
#> [1] "double"
```

## Character (Texto/String)

```
# Sempre entre aspas (simples ' ou duplas ")
nome <- "Maria Silva"
cidade <- 'S o Paulo'
curso <- "R Programming"
```

```
class(nome)
```

```
#> [1] "character"
```

```
# Concatenar strings
```

```
paste("01 ", nome)
```

```
#> [1] "01 , Maria Silva"
```

```
paste0("01 ", nome) # Sem espa o autom tico
```

```
#> [1] "01 , Maria Silva"
```

## Logical (L gico)

```
# Apenas TRUE ou FALSE (sempre mai sculas)
aprovado <- TRUE
reprovado <- FALSE
tem_dados <- TRUE
```

```
class(aprovado)
```

```
#> [1] "logical"
```

```
# Atalhos: T para TRUE, F para FALSE (mas evite, menos claro)
```

```
x <- T
```

## Factor (Fator/Categ rico)

```
# Para vari veis categ ricas com n veis fixos
sexo <- factor(c("M", "F", "F", "M", "M"))
escolaridade <- factor(
  c("Fundamental", "M dio", "Superior", "M dio"),
  levels = c("Fundamental", "M dio", "Superior"),
  ordered = TRUE # Tem ordem
)
```

```
class(sexo)
```

```
#> [1] "factor"
```

```
levels(sexo)
```

```
#> [1] "F" "M"
```

## Conversões entre Tipos

```
# Numeric para character
x <- 42
y <- as.character(x)
y
```

```
#> [1] "42"
```

```
class(y)
```

```
#> [1] "character"
```

```
# Character para numeric
texto <- "3.14"
numero <- as.numeric(texto)
numero
```

```
#> [1] 3.14
```

```
# Logical para numeric (TRUE = 1, FALSE = 0)
as.numeric(TRUE)
```

```
#> [1] 1
```

```
as.numeric(FALSE)
```

```
#> [1] 0
```

```
# Cuidado com conversões impossíveis!
as.numeric("abc") # Retorna NA (missing)
```

```
#> [1] NA
```

---

## 2.4 Vetores

**Vetor** a estrutura de dados mais básica do R. uma sequência de elementos **do mesmo tipo**.

### Criando Vetores

```
# Função c() = combine
idades <- c(23, 45, 19, 34, 28)
nomes <- c("Ana", "Bruno", "Carla", "Diego", "Elena")
aprovados <- c(TRUE, TRUE, FALSE, TRUE, TRUE)
```

```
# Sequências
1:10 # 1, 2, 3, ..., 10
```

```
#> [1] 1 2 3 4 5 6 7 8 9 10
```

```
10:1 # Decrescente
```

```
#> [1] 10 9 8 7 6 5 4 3 2 1
```

```
seq(0, 100, by = 10) # 0, 10, 20, ..., 100
```

```
#> [1] 0 10 20 30 40 50 60 70 80 90 100
```

```
seq(0, 1, length.out = 5) # 5 n meros entre 0 e 1
```

```
#> [1] 0.00 0.25 0.50 0.75 1.00
```

```
# Repeti es  
rep(5, times = 3) # 5, 5, 5
```

```
#> [1] 5 5 5
```

```
rep(c(1, 2), times = 3) # 1, 2, 1, 2, 1, 2
```

```
#> [1] 1 2 1 2 1 2
```

```
rep(c(1, 2), each = 3) # 1, 1, 1, 2, 2, 2
```

```
#> [1] 1 1 1 2 2 2
```

### Propriedades de Vetores

```
idades <- c(23, 45, 19, 34, 28)
```

```
length(idades) # Tamanho
```

```
#> [1] 5
```

```
class(idades) # Tipo
```

```
#> [1] "numeric"
```

```
typeof(idades) # Tipo interno
```

```
#> [1] "double"
```

```
# Estat sticas descritivas  
mean(idades) # M dia
```

```
#> [1] 29.8
```

```
median(idades) # Mediana
```

```
#> [1] 28
```

```
sd(idades) # Desvio padr o
```

```
#> [1] 10.18332
```

```
min(idades) # M nimo
```

```
#> [1] 19
```

```
max(idades) # M ximo
```

```
#> [1] 45
```

```
sum(idades) # Soma
```

```
#> [1] 149
```

```
range(idades) # M nimo e m ximo
```

```
#> [1] 19 45
```

## Operações Vetorializadas

O R opera em vetores inteiros automaticamente!

```
# Aritméticas
x <- c(1, 2, 3, 4, 5)
x + 10          # Soma 10 a cada elemento

#> [1] 11 12 13 14 15

x * 2           # Multiplica cada elemento por 2

#> [1]  2  4  6  8 10

x^2            # Eleva cada elemento ao quadrado

#> [1]  1  4  9 16 25

# Entre vetores
y <- c(10, 20, 30, 40, 50)
x + y          # Soma elemento por elemento

#> [1] 11 22 33 44 55

x * y          # Multiplica elemento por elemento

#> [1] 10 40 90 160 250

# Exemplo prático: IMC
peso <- c(70, 85, 62, 90, 75)
altura <- c(1.75, 1.80, 1.65, 1.78, 1.82)
imc <- peso / altura^2
imc

#> [1] 22.85714 26.23457 22.77319 28.40550 22.64219
```

---

## 2.5 Indexação de Vetores

Indexação = acessar elementos específicos de um vetor.

Por Posição

```
nomes <- c("Ana", "Bruno", "Carla", "Diego", "Elena")

# Primeiro elemento (R começa em 1, não em 0!)
nomes[1]

#> [1] "Ana"

# Terceiro elemento
nomes[3]

#> [1] "Carla"

# Múltiplos elementos
nomes[c(1, 3, 5)] # Posições 1, 3 e 5

#> [1] "Ana" "Carla" "Elena"

# Sequência
nomes[2:4] # Do 2 ao 4
```

```
#> [1] "Bruno" "Carla" "Diego"
# ltimo elemento
nomes[length(nomes)]

#> [1] "Elena"
# Todos menos o primeiro
nomes[-1]

#> [1] "Bruno" "Carla" "Diego" "Elena"
# Todos menos o 2 e 4
nomes[-c(2, 4)]

#> [1] "Ana" "Carla" "Elena"
```

## Por Condição Lógica

```
idades <- c(23, 45, 19, 34, 28)

# Quem tem mais de 25 anos?
idades > 25

#> [1] FALSE TRUE FALSE TRUE TRUE
# Pegar apenas quem tem mais de 25
idades[idades > 25]

#> [1] 45 34 28
# Múltiplas condições (& = E, | = OU)
idades[idades > 20 & idades < 40] # Entre 20 e 40

#> [1] 23 34 28
idades[idades < 20 | idades > 40] # Menor que 20 OU maior que 40

#> [1] 45 19
# Exemplo prático
nomes <- c("Ana", "Bruno", "Carla", "Diego", "Elena")
idades <- c(23, 45, 19, 34, 28)

# Nomes de quem tem mais de 25 anos
nomes[idades > 25]

#> [1] "Bruno" "Diego" "Elena"
```

## Por Nome

```
# Vetores podem ter nomes
notas <- c(ana = 8.5, bruno = 7.0, carla = 9.5)
notas

#> ana bruno carla
#> 8.5 7.0 9.5
```

```

# Acessar por nome
notas["ana"]

#> ana
#> 8.5

notas[c("ana", "carla")]

#>   ana carla
#> 8.5   9.5

# Ver os nomes
names(notas)

#> [1] "ana"   "bruno" "carla"

# Adicionar nomes depois
idades <- c(23, 45, 19)
names(idades) <- c("Ana", "Bruno", "Carla")
idades

#>   Ana Bruno Carla
#>   23   45   19

```

---

## 2.6 Valores Ausentes (NA)

NA = Not Available (valor ausente/faltante)

```

# Criar vetor com NA
alturas <- c(1.75, NA, 1.82, 1.68, NA)

# Identificar NAs
is.na(alturas)

#> [1] FALSE  TRUE FALSE FALSE  TRUE

# Contar quantos NAs
sum(is.na(alturas))

#> [1] 2

# Funções com NA precisam de na.rm = TRUE
mean(alturas)           # Retorna NA

#> [1] NA

mean(alturas, na.rm = TRUE) # Ignora NAs

#> [1] 1.75

# Remover NAs
alturas[!is.na(alturas)] # != NOT (negação)

#> [1] 1.75 1.82 1.68

na.omit(alturas)

#> [1] 1.75 1.82 1.68
#> attr(,"na.action")
#> [1] 2 5

```

```
#> attr("class")
#> [1] "omit"
```

---

## 2.7 Listas

Listas podem conter elementos de **tipos diferentes** (vetores s um tipo).

```
# Criar lista
pessoa <- list(
  nome = "Ana Silva",
  idade = 28,
  altura = 1.68,
  casado = FALSE,
  filhos = c("Jo o", "Maria")
)

# Ver estrutura
str(pessoa)
```

```
#> List of 5
#> $ nome : chr "Ana Silva"
#> $ idade : num 28
#> $ altura: num 1.68
#> $ casado: logi FALSE
#> $ filhos: chr [1:2] "Jo o" "Maria"
```

```
# Acessar elementos (3 formas)
pessoa$nome # Pelo nome (mais comum)
```

```
#> [1] "Ana Silva"
```

```
pessoa[["nome"]] # Pelo nome (alternativa)
```

```
#> [1] "Ana Silva"
```

```
pessoa[[1]] # Pela posição
```

```
#> [1] "Ana Silva"
```

```
# Acessar sub-elementos
pessoa$filhos[1] # Primeiro filho
```

```
#> [1] "Jo o"
```

---

## 2.8 Data Frames

**Data frame** = tabela de dados (como Excel, mas melhor!)

```
# Criar data frame
alunos <- data.frame(
  nome = c("Ana", "Bruno", "Carla", "Diego"),
  idade = c(23, 25, 22, 24),
  nota = c(8.5, 7.0, 9.5, 8.0),
  aprovado = c(TRUE, TRUE, TRUE, TRUE)
)
```

```
# Ver dados
```

```
alunos
```

```
#>   nome idade nota aprovado
#> 1  Ana    23  8.5      TRUE
#> 2 Bruno   25  7.0      TRUE
#> 3 Carla   22  9.5      TRUE
#> 4 Diego   24  8.0      TRUE
```

```
print(alunos)
```

```
#>   nome idade nota aprovado
#> 1  Ana    23  8.5      TRUE
#> 2 Bruno   25  7.0      TRUE
#> 3 Carla   22  9.5      TRUE
#> 4 Diego   24  8.0      TRUE
```

```
# Estrutura
```

```
str(alunos)
```

```
#> 'data.frame':   4 obs. of  4 variables:
#> $ nome      : chr  "Ana" "Bruno" "Carla" "Diego"
#> $ idade     : num  23 25 22 24
#> $ nota      : num  8.5 7 9.5 8
#> $ aprovado: logi  TRUE TRUE TRUE TRUE
```

```
# Primeiras linhas
```

```
head(alunos, 2) # Primeiras 2 linhas
```

```
#>   nome idade nota aprovado
#> 1  Ana    23  8.5      TRUE
#> 2 Bruno   25  7.0      TRUE
```

```
# Últimas linhas
```

```
tail(alunos, 2)
```

```
#>   nome idade nota aprovado
#> 3 Carla   22  9.5      TRUE
#> 4 Diego   24  8.0      TRUE
```

```
# Dimensões
```

```
nrow(alunos) # Número de linhas
```

```
#> [1] 4
```

```
ncol(alunos) # Número de colunas
```

```
#> [1] 4
```

```
dim(alunos) # Ambos
```

```
#> [1] 4 4
```

```
# Nomes das colunas
```

```
names(alunos)
```

```
#> [1] "nome" "idade" "nota" "aprovado"
```

```
colnames(alunos)
```

```
#> [1] "nome" "idade" "nota" "aprovado"
```



## Acessar Colunas

```
# Por nome ($) - mais comum
alunos$nome

#> [1] "Ana" "Bruno" "Carla" "Diego"

alunos$nota

#> [1] 8.5 7.0 9.5 8.0

# Por posição
alunos[, 3] # Terceira coluna

#> [1] 8.5 7.0 9.5 8.0

alunos[, "nota"] # Por nome (alternativa)

#> [1] 8.5 7.0 9.5 8.0

# Múltiplas colunas
alunos[, c("nome", "nota")]

#>   nome nota
#> 1  Ana  8.5
#> 2 Bruno  7.0
#> 3 Carla  9.5
#> 4 Diego  8.0

alunos[, c(1, 3)]

#>   nome nota
#> 1  Ana  8.5
#> 2 Bruno  7.0
#> 3 Carla  9.5
#> 4 Diego  8.0
```

## Acessar Linhas

```
# Por posição
alunos[1, ] # Primeira linha (todas as colunas)

#>   nome idade nota aprovado
#> 1  Ana    23  8.5      TRUE

alunos[2:3, ] # Linhas 2 e 3

#>   nome idade nota aprovado
#> 2 Bruno    25  7.0      TRUE
#> 3 Carla    22  9.5      TRUE

# Por condição
alunos[alunos$nota > 8, ] # Quem tirou mais de 8

#>   nome idade nota aprovado
#> 1  Ana    23  8.5      TRUE
#> 3 Carla    22  9.5      TRUE

alunos[alunos$idade < 24, ] # Quem tem menos de 24 anos

#>   nome idade nota aprovado
```

```
#> 1   Ana    23  8.5    TRUE
#> 3 Carla    22  9.5    TRUE
```

```
alunos[alunos$nome == "Ana", ]      # Linha da Ana
```

```
#>   nome idade nota aprovado
#> 1   Ana    23  8.5    TRUE
```

## Acessar C lulas Espec ficas

```
# [linha, coluna]
alunos[1, 2]      # Linha 1, coluna 2 (idade da Ana)
```

```
#> [1] 23
```

```
alunos[2, "nota"]  # Nota do Bruno
```

```
#> [1] 7
```

```
alunos[1:2, c(1, 3)] # Linhas 1-2, colunas nome e nota
```

```
#>   nome nota
#> 1   Ana  8.5
#> 2 Bruno 7.0
```

## Adicionar Colunas

```
# Criar nova coluna
alunos$frequencia <- c(95, 87, 100, 92)
alunos
```

```
#>   nome idade nota aprovado frequencia
#> 1   Ana    23  8.5    TRUE         95
#> 2 Bruno    25  7.0    TRUE         87
#> 3 Carla    22  9.5    TRUE        100
#> 4 Diego    24  8.0    TRUE         92
```

```
# Colunas calculadas
alunos$nota_ajustada <- alunos$nota * 1.1 # Aumenta 10%
alunos
```

```
#>   nome idade nota aprovado frequencia nota_ajustada
#> 1   Ana    23  8.5    TRUE         95         9.35
#> 2 Bruno    25  7.0    TRUE         87         7.70
#> 3 Carla    22  9.5    TRUE        100        10.45
#> 4 Diego    24  8.0    TRUE         92         8.80
```

## 2.9 Fun es teis de Explora o

```
# Criar dataset de exemplo
dados <- data.frame(
  id = 1:5,
  nome = c("Ana", "Bruno", "Carla", "Diego", "Elena"),
  idade = c(23, 45, 19, 34, 28),
  altura = c(1.65, 1.80, 1.58, 1.75, 1.70),
  peso = c(58, 85, 52, 78, 65)
```

```

)

# Resumo estatístico
summary(dados)

#>      id      nome      idade      altura      peso
#> Min.   :1   Length:5      Min.   :19.0   Min.   :1.580   Min.   :52.0
#> 1st Qu.:2   Class :character 1st Qu.:23.0   1st Qu.:1.650   1st Qu.:58.0
#> Median :3   Mode  :character  Median :28.0   Median :1.700   Median :65.0
#> Mean   :3                                Mean   :29.8   Mean   :1.696   Mean   :67.6
#> 3rd Qu.:4                                3rd Qu.:34.0   3rd Qu.:1.750   3rd Qu.:78.0
#> Max.   :5                                Max.   :45.0   Max.   :1.800   Max.   :85.0

# Estrutura detalhada
str(dados)

#> 'data.frame': 5 obs. of 5 variables:
#> $ id : int 1 2 3 4 5
#> $ nome : chr "Ana" "Bruno" "Carla" "Diego" ...
#> $ idade : num 23 45 19 34 28
#> $ altura: num 1.65 1.8 1.58 1.75 1.7
#> $ peso : num 58 85 52 78 65

# Glimpse (tidyverse) - mais compacto
dplyr::glimpse(dados)

#> Rows: 5
#> Columns: 5
#> $ id <int> 1, 2, 3, 4, 5
#> $ nome <chr> "Ana", "Bruno", "Carla", "Diego", "Elena"
#> $ idade <dbl> 23, 45, 19, 34, 28
#> $ altura <dbl> 1.65, 1.80, 1.58, 1.75, 1.70
#> $ peso <dbl> 58, 85, 52, 78, 65

# Primeiras/ últimas linhas
head(dados, 3)

#>   id nome idade altura peso
#> 1  1  Ana   23   1.65   58
#> 2  2 Bruno  45   1.80   85
#> 3  3 Carla  19   1.58   52

tail(dados, 2)

#>   id nome idade altura peso
#> 4  4 Diego  34   1.75   78
#> 5  5 Elena  28   1.70   65

# Ver dados em planilha (não usar em scripts - só interativo)
# View(dados)

```

## EXERCICIO: Exercícios Práticos

### Exercício 1: Calculadora e Objetos

```
# a) Calcule: (15 + 23) * 4 / 2

# b) Crie objetos para seu nome, idade e cidade

# c) Use paste() para criar a frase: "Meu nome [nome], tenho [idade] anos e moro em [cidade]"

# d) Calcule quantos meses você tem de vida (idade * 12)
```

### Exercício 2: Vetores

```
# a) Crie um vetor com as temperaturas da semana: 25, 27, 23, 26, 28, 30, 29

# b) Calcule a temperatura média

# c) Quais dias tiveram temperatura acima de 26?

# d) Crie um vetor com os dias da semana e combine com as temperaturas
```

### Exercício 3: Data Frames

```
# a) Crie um data frame com informações de 5 amigos:
#     - nome, idade, cidade, profissão

# b) Mostre apenas os nomes

# c) Filtre apenas quem tem mais de 25 anos

# d) Adicione uma coluna "salario" com valores fictícios
```

### Exercício 4: Análise Real

```
# Dataset: pacientes de uma clínica
pacientes <- data.frame(
  id = 1:10,
  idade = c(34, 45, 23, 56, 41, 29, 38, 52, 31, 47),
  peso = c(70, 85, 58, 92, 75, 63, 80, 88, 65, 78),
  altura = c(1.68, 1.75, 1.60, 1.82, 1.70, 1.65, 1.78, 1.80, 1.63, 1.73),
  pressao_alta = c(F, T, F, T, T, F, F, T, F, T)
)
```

```
# a) Calcule o IMC de cada paciente (peso / altura^2)

# b) Quantos pacientes têm hipertensão?

# c) Qual a média de idade dos hipertensos vs. não hipertensos?

# d) Crie uma categoria de IMC: "Baixo" (<18.5), "Normal" (18.5-24.9),
#     "Sobrepeso" (25-29.9), "Obesidade" (>=30)
```

---

## OBJETIVO: Prática Guiada: Primeiro Script

Vamos criar nosso primeiro script completo!

### Passo 1: Criar Script

No RStudio: - **File New File R Script** (Ctrl + Shift + N) - Salve como: scripts/01\_fundamentos.R

### Passo 2: Escrever Código

```
# =====
# Script: Análise Exploratória - Fundamentos
# Autor: Seu Nome
# Data: 2025-01-XX
# Descrição: Primeiro script do curso - análise de dados fictícios de saúde
# =====

# Carregar pacotes ----
library(tidyverse)
library(here)

# Criar dados ----
# Dados fictícios de uma clínica
dados_clinica <- data.frame(
  paciente_id = 1:20,
  idade = sample(20:70, 20, replace = TRUE),
  sexo = sample(c("M", "F"), 20, replace = TRUE),
  peso = rnorm(20, mean = 70, sd = 15),
  altura = rnorm(20, mean = 1.70, sd = 0.10),
  pressao_sistolica = rnorm(20, mean = 120, sd = 15),
  fumante = sample(c(TRUE, FALSE), 20, replace = TRUE)
)

# Explorar dados ----
glimpse(dados_clinica)
summary(dados_clinica)

# Criar variáveis derivadas ----
```

```

dados_clinica$imc <- dados_clinica$peso / dados_clinica$altura^2
dados_clinica$hipertenso <- dados_clinica$pressao_sistolica >= 140

# Análises simples ----
# Média de idade
mean(dados_clinica$idade)

# Proporção de fumantes
mean(dados_clinica$fumante) # TRUE = 1, FALSE = 0

# IMC médio por sexo
tapply(dados_clinica$imc, dados_clinica$sexo, mean)

# Quantos hipertensos?
table(dados_clinica$hipertenso)

# Exportar dados processados ----
# (vamos aprender mais sobre isso no Dia 4)
# write_csv(dados_clinica, here("data", "processed", "dados_clinica_processados.csv"))

# Fim do script ----

```

### Passo 3: Executar

- **Linha por linha:** Ctrl + Enter
- **Tudo:** Ctrl + Shift + Enter
- **At a linha atual:** Ctrl + Alt + B

## GITHUB: Primeiro Commit no GitHub

Agora vamos versionar nosso trabalho!

### Via RStudio (Recomendado)

1. **Aba Git** (canto superior direito)
2. Marque [OK] os arquivos modificados
3. Clique em **Commit**
4. Escreva mensagem: "Dia 1: adiciona fundamentos e primeiro script"
5. Clique **Commit**
6. Clique **Push**

### Via Terminal

```

# Ver o que mudou
git status

# Adicionar arquivos
git add .

# Commit
git commit -m "Dia 1: adiciona fundamentos e primeiro script"

```

```
# Enviar para GitHub  
git push
```

## Verificar

Abra seu repositório no GitHub e veja os arquivos lá!

---

## MATERIAL: Para Casa

1. **Revisar** todo o material do Dia 1
  2. **Refazer** os exercícios sem consultar as respostas
  3. **Explorar** os cheat sheets (Help Cheat Sheets no RStudio)
  4. **Experimentar** com seus próprios dados (se tiver)
  5. **Ler** capítulos 1-3 do R for Data Science
- 

## Recursos Adicionais

### Documentação

- R for Data Science (2e) - Capítulos 1-4
- Hands-On Programming with R
- RStudio Cheat Sheets

### Prática

- Swirl - Aprenda R dentro do R
- R-exercises - Exercícios práticos

### Comunidades

- Posit Community
  - Stack Overflow [r]
  - R-Ladies
- 

## Dúvidas Frequentes

### P: Por que usar `<-` em vez de `=` para atribuição?

R: Convenção histórica do R. Ambos funcionam, mas `<-` mais idiomático.

### P: Por que meu código não funciona?

R: Principais causas: (1) parênteses/aspas não fechados, (2) vírgulas faltando, (3) objeto não existe (typo no nome), (4) pacote não carregado.

### P: Devo usar `library()` ou `require()`?

R: Use `library()`. Ela dá erro se o pacote não existe (o que bom!).

### P: Como limpar o console?

R: Ctrl + L (mas isso não remove objetos, só limpa visualmente).

**P: Como remover objetos da memória?**

R: `rm(nome_objeto)` ou `rm(list = ls())` para remover tudo.

---

## CURSO: Conclusão do Dia 1

Parabéns! Você completou o Dia 1 e agora sabe:

- [OK] Configurar ambiente completo (R, RStudio, Git, GitHub)
- [OK] Tipos de dados e estruturas fundamentais
- [OK] Criar e manipular vetores e data frames
- [OK] Indexar e filtrar dados
- [OK] Organizar projetos e usar caminhos relativos
- [OK] Fazer commits no GitHub

**Amanhã:** Lógica de programação, funções customizadas e uso de IA!

---

\*\* Última atualização: \*\* 2025-10-07

**Contato:** [seu-email@example.com](mailto:seu-email@example.com)

**Repositório:** <https://github.com/seu-usuario/curso-r-github-ia>