Dia 5: Relatórios Reprodutíveis e Projeto Final

Curso: Introdução à Programação em R com GitHub e IA

Vinícius Silva Junqueira

2025-10-07

Contents

1 Apresenta o do Curso

1.1 Informa es Gerais

Carga hor ria total: 16 horas
Per odo: Segunda a sexta-feira
Hor rio: 19h00 s 22h00 (intervalo de 20 min s 20h30)
P blico-alvo: Iniciantes de diversas reas

1.2 Objetivos do Curso

Ao final deste curso, voc ser capaz de:

- Realizar an lises de dados usando R e tidyverse
- Criar visualiza es profissionais com ggplot2
- Versionar c digo e colaborar usando Git/GitHub
- Produzir relat rios reprodut veis com RMarkdown
- Utilizar IA (ChatGPT e Claude) de forma estrat gica
- Organizar projetos de an lise de dados
- Aplicar boas pr ticas de programa o

1.3 Estrutura do Curso

1.3.1 Dia 1: Fundamentos e Ambiente Reprodut vel

Setup completo, tipos de dados, vetores, data frames, projetos RStudio

1.3.2 Dia 2: L gica e Programa o

Operadores, condicionais, loops, fun es, boas pr ticas, debugging, IA

1.3.3 Dia 3: Manipula o de Dados

Tidyverse, dplyr, tidyr, tratamento de NAs, janitor, skimr

1.3.4 Dia 4: I/O e Visualiza o

Leitura/escrita de dados, organiza o de projetos, ggplot2

1.3.5 Dia 5: Relat rios e Projeto Final

RMarkdown, documentos reprodut veis, projeto final, GitHub

1.4 Pacotes Necess rios

```
install.packages(c(
  "tidyverse", "here", "janitor", "skimr",
  "readxl", "writexl", "rmarkdown",
  "usethis", "gitcreds"
))
```

2 Parte 1: Fundamentos e Ambiente Reprodut vel

---

3 OBJETIVO: Objetivos do Dia 1

Ao final desta aula, voc ser 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 opera es b sicas de indexa o
[OK] Organizar projetos com RStudio Projects
[OK] Fazer seu primeiro commit no GitHub
[OK] Usar o pacote `here` para caminhos relativos

---

4 MATERIAL: Parte 1: Ambienta o e Setup (19h00 - 20h30)

4.1 1.1 Por que R, GitHub e IA?

4.1.1 Por que R?

R uma linguagem de programa o estat stica amplamente utilizada em:

- `**Pesquisa cient  fica:** an  lise de dados experimentais`
- `**Sa  de p  blica:** epidemiologia, ensaios cl  nicos`
- GRAFICO: **Economia e finan as:** modelagem econom trica, s ries temporais
- `**Ecologia:** an  lise de biodiversidade, modelos populacionais`
- `**Marketing e vendas:** segmenta  o, an  lise de campanhas`
- `**Ci  ncias sociais:** an  lise de surveys, dados censit  rios`

Vantagens: - **Gratuito e open-source** - **Comunidade ativa** e acolhedora (R-Ladies, Stack Overflow) - **Pacotes especializados** para praticamente qualquer an lise - **Reprodutibilidade:** RMarkdown permite documentar an lises - **Visualiza es de alta qualidade:** ggplot2 refer ncia mundial

4.1.2 Por que GitHub?

GitHub uma plataforma de versionamento de c digo que permite:

- NOTA: **Hist rico completo** de todas as mudan as
- WORKFLOW: **Colabora o** eficiente em equipe
- `**Backup autom  tico** na nuvem`
- `**Portf  lio p  blico** de projetos`
- `**Compartilhamento** f  cil de an  lises`

4.1.3 Por que IA (ChatGPT e Claude)?

Ferramentas de IA generativa aceleram o aprendizado:

- DICA: **Explica es personalizadas** de conceitos dif ceis
- `**Debugging assistido:** entender erros rapidamente`
- NOTA: **Documenta o autom tica:** comentar c digo
- INICIO: **Produtividade:** sugest es de c digo e otimiza es
- CURSO: **Tutor 24/7:** responde d vidas a qualquer momento

  ATENCAO: **Aten o:** IA uma ferramenta auxiliar. Sempre entenda o c digo antes de usar!

4.2 1.2 Verifica o das Instala es

Antes de come ar, vamos verificar se tudo est instalado corretamente.

4.2.1 Verificar R

```
# Vers  o 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
```

4.2.2 Verificar RStudio

No menu: **Help About RStudio**
Vers o recomendada: 2023.09 ou superior

4.2.3 Verificar Git

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

```
git --version
```

Deve retornar algo como: `git version 2.40.0`

Se n o 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)

4.3 1.3 Instala o de Pacotes Essenciais

Vamos instalar os pacotes que usaremos durante o curso:

```r
# Instalar pacotes (execute apenas uma vez)
install.packages(c(
  "tidyverse",    # Conjunto de pacotes para ci  ncia de dados
  "here",         # Caminhos relativos seguros
  "janitor",      # Limpeza de dados
  "skimr",        # Resumos estat  sticos
  "readxl",       # Leitura de arquivos Excel
  "writexl",      # Escrita de arquivos Excel
  "rmarkdown",    # Relat  rios reprodut  veis
  "usethis",      # Ferramentas de desenvolvimento
  "gitcreds"      # Gerenciamento de credenciais Git
))
```

4.3.1 Carregar pacotes

```r
# 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!

4.4 1.4 Configura o do GitHub

4.4.1 Criar Conta no GitHub

Se ainda n o tem: https://github.com/signup

Dicas: - Use um email profissional/acad mico - Username curto e profissional (seu nome ou varia o) - Ative autentica o de dois fatores (2FA)

4.4.2 Configurar Git Local

```r
# 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"
```

4.4.3 Criar Personal Access Token (PAT)

O PAT como uma senha especial para o Git:

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

No GitHub: 1. D um nome descritivo (ex: "Curso R 2025") 2. Selecione expira o (90 dias ou mais) 3. Marque apenas: `repo`, `workflow`, `gist` 4. Clique em "Generate token" 5. **COPIE O TOKEN** (n o conseguir ver novamente!)

4.4.4 Salvar Token Localmente

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

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

4.5 1.5 Clonando o Reposit rio do Curso

4.5.1 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**

4.5.2 Via Terminal

```bash
# Navegue at   onde quer salvar
cd ~/Documents

# Clone o reposit  rio
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 d   duplo clique no .Rproj)
```

4.6 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                 # Descri    o 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/                # Gr  ficos salvos
      tables/                 # Tabelas exportadas

materiais/                  # Material do curso
      dia1_fundamentos.Rmd
      ...

docs/                       # Relat  rios finais
```

4.6.1 Por que usar Projetos RStudio?

[OK] **Working directory autom tico:** sempre aponta para a pasta do projeto
[OK] **Portabilidade:** funciona em qualquer computador
[OK] **Organiza o:** mant m tudo junto
[OK] **Integra o com Git:** painel Git aparece automaticamente

---

4.7 1.7 O Pacote here

O `here` resolve problemas de caminhos de arquivos:

```
# Caminho absoluto (RUIM - n  o 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
```

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

Sempre use `here()` para referenciar arquivos!

---

5 INTERVALO: INTERVALO (20h30 - 20h50)

Aproveite para: - Tomar gua/caf
- Conferir se todas as instala es funcionaram - Tirar d vidas no grupo

---

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

6.1 2.1 R como Calculadora

O jeito mais simples de come ar: usar R como calculadora avan ada!

```
# Opera    es b  sicas
2 + 3        # Adi    o
```

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

```
10 - 4       # Subtra    o
```

```
#> [1] 6
```

```
5 * 6        # Multiplica    o
```

```
#> [1] 30
```

```
20 / 4       # Divis  o
```

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

```
2^3          # Pot  ncia
```

```
#> [1] 8
```

```
sqrt(16)     # Raiz quadrada
```

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

```
log(10)      # Logaritmo natural
```

```
#> [1] 2.302585
```

```
log10(100)   # Logaritmo base 10
```

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

```
# Ordem de opera    es (PEMDAS)
2 + 3 * 4        # 14, n  o 20!
```

```
#> [1] 14
```

```
(2 + 3) * 4      # 20, com par  nteses
```

```
#> [1] 20
```

6.2 2.2 Objetos e Atribui o

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

```
# Criar objetos
x <- 5
y <- 10
nome <- "Vin  cius"
aprovado <- TRUE

# Ver conte  do
x
```

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

```
print(y)
```

```
#> [1] 10
```

```
# Usar em opera    es
resultado <- x + y
resultado
```

```
#> [1] 15
```

```
# Sobrescrever
x <- 20
x  # Agora vale 20, n  o mais 5!
```

```
#> [1] 20
```

6.2.1 Regras para Nomes de Objetos

[OK] **Permitido:**

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

[X] **N O permitido:**

```
2idade <- 30        # N  o pode come  ar com n  mero
idade-media <- 25   # H  fen n  o    permitido (use _)
minha idade <- 30   # Sem espa  os!
```

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

6.3 2.3 Tipos de Dados Fundamentais

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

6.3.1 Numeric (N meros)

```
# 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"
```

6.3.2 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("Ol  ,", nome)
```

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

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

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

6.3.3 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
```

6.3.4 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"
```

6.3.5 Convers o 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 convers  es imposs  veis!
as.numeric("abc")  # Retorna NA (missing)
```

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

6.4 2.4 Vetores

Vetor a estrutura de dados mais b sica do R. uma sequ ncia de elementos **do mesmo tipo**.

6.4.1 Criando Vetores

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

# Sequ  ncias
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
```

6.4.2 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
```

6.4.3 Opera es Vetorizadas

O R opera em vetores inteiros automaticamente!

```
# Aritm  ticas
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
```

```r
# Exemplo pr  tico: 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
```

6.5 2.5 Indexa o de Vetores

Indexa o = acessar elementos espec ficos de um vetor.

6.5.1 Por Posi o

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

# Primeiro elemento (R come  a em 1, n  o em 0!)
nomes[1]
```

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

```r
# Terceiro elemento
nomes[3]
```

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

```r
# M  ltiplos elementos
nomes[c(1, 3, 5)]  # Posi    es 1, 3 e 5
```

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

```r
# Sequ  ncia
nomes[2:4]  # Do 2   ao 4
```

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

```r
#   ltimo elemento
nomes[length(nomes)]
```

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

```r
# Todos menos o primeiro
nomes[-1]
```

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

```r
# Todos menos o 2   e 4
nomes[-c(2, 4)]
```

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

6.5.2 Por Condi o L gica

```
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
```

```
# M  ltiplas condi    es (& = 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 pr  tico
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"
```

6.5.3 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
```

6.6 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
```

```
# Fun    es 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 (nega    o)
```

```
#> [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"
```

6.7 2.7 Listas

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

```r
# 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"
```

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

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

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

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

```r
pessoa[[1]]           # Pela posi    o
```

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

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

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

6.8 2.8 Data Frames

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

```r
# 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)
)
```

```r
# 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
```

```r
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
```

```r
# 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
```

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

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

```r
#   ltimas linhas
tail(alunos, 2)
```

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

```r
# Dimens  es
nrow(alunos)     # N  mero de linhas
```

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

```r
ncol(alunos)     # N  mero de colunas
```

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

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

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

```r
# Nomes das colunas
names(alunos)
```

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

```r
colnames(alunos)
```

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

6.8.1 Acessar Colunas

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

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

```r
alunos$nota
```

```
#> [1] 8.5 7.0 9.5 8.0
```

```r
# Por posi    o
alunos[, 3]      # Terceira coluna
```

```
#> [1] 8.5 7.0 9.5 8.0
```

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

```
#> [1] 8.5 7.0 9.5 8.0
```

```r
# M  ltiplas colunas
alunos[, c("nome", "nota")]
```

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

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

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

6.8.2 Acessar Linhas

```r
# Por posi    o
alunos[1, ]      # Primeira linha (todas as colunas)
```

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

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

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

```r
# Por condi    o
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
```

```r
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
```

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

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

6.8.3 Acessar C lulas Espec ficas

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

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

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

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

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

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

6.8.4 Adicionar Colunas

```r
# 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
```

```r
# 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
```

6.9 2.9 Fun es teis de Explora o

```r
# 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 estat  stico
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
```

```r
# 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
```

```r
# 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/  ltimas 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 (n  o usar em scripts - s   interativo)
# View(dados)
```

---

7 EXERCICIO: Exerc cios Pr ticos

7.1 Exerc cio 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 voc   tem de vida (idade * 12)
```

7.2 Exerc cio 2: Vetores

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


# b) Calcule a temperatura m  dia


# c) Quais dias tiveram temperatura acima de 26?
```

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

7.3 Exerc cio 3: Data Frames

```r
# a) Crie um data frame com informa    es de 5 amigos:
#    - nome, idade, cidade, profiss  o


# b) Mostre apenas os nomes


# c) Filtre apenas quem tem mais de 25 anos


# d) Adicione uma coluna "salario" com valores fict  cios
```

7.4 Exerc cio 4: An lise Real

```r
# Dataset: pacientes de uma cl  nica
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 t  m hipertens  o?


# c) Qual a m  dia de idade dos hipertensos vs. n  o hipertensos?


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

---

8 OBJETIVO: Pr tica Guiada: Primeiro Script

Vamos criar nosso primeiro script completo!

8.1 Passo 1: Criar Script

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

8.2 Passo 2: Escrever C digo

```
# ==============================================================================
# Script: An  lise Explorat  ria - Fundamentos
# Autor: Seu Nome
# Data: 2025-01-XX
# Descri    o: Primeiro script do curso - an  lise de dados fict  cios de sa  de
# ==============================================================================

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

# Criar dados ----
# Dados fict  cios de uma cl  nica
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 vari  veis derivadas ----
dados_clinica$imc <- dados_clinica$peso / dados_clinica$altura^2
dados_clinica$hipertenso <- dados_clinica$pressao_sistolica >= 140

# An  lises simples ----
# M  dia de idade
mean(dados_clinica$idade)

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

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

# Quantos hipertensos?
```

```r
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 ----
```

8.3 Passo 3: Executar

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

---

9 GITHUB: Primeiro Commit no GitHub

Agora vamos versionar nosso trabalho!

9.1 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**

9.2 Via Terminal

```bash
# 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
```

9.3 Verificar

Abra seu reposit rio no GitHub e veja os arquivos l !

---

10 MATERIAL: Para Casa

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

---

11 Recursos Adicionais

11.1 Documenta o

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

11.2 Pr tica

- Swirl - Aprenda R dentro do R
- R-exercises - Exerc cios pr ticos

11.3 Comunidades

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

---

12 D vidas Frequentes

P: Por que usar `<-` em vez de `=` para atribuição?
R: Conven o hist rica do R. Ambos funcionam, mas `<-` mais idiom tico.

P: Por que meu c digo n o funciona?
R: Principais causas: (1) par nteses/aspas n o fechados, (2) v rgulas faltando, (3) objeto n o existe (typo no nome), (4) pacote n o carregado.

P: Devo usar `library()` ou `require()`?
R: Use `library()`. Ela d erro se o pacote n o existe (o que bom!).

P: Como limpar o console?
R: Ctrl + L (mas isso n o remove objetos, s limpa visualmente).

P: Como remover objetos da mem ria?
R: `rm(nome_objeto)` ou `rm(list = ls())` para remover tudo.

---

13 CURSO: Conclus o do Dia 1

Parab ns! Voc 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

Amanh : L gica de programa o, fun es customizadas e uso de IA!

** ltima atualiza o:** 2025-10-07
Contato: seu-email@example.com
Reposit rio: https://github.com/seu-usuario/curso-r-github-ia

14 Parte 2: L gica e Programa o

---

15 Objetivos do Dia 2

Ao final desta aula, você será capaz de:

- Usar operadores relacionais e lógicos para comparações
- Criar estruturas condicionais (if, else, ifelse, case_when)
- Entender quando e como usar loops
- Criar funções customizadas para automatizar tarefas
- Aplicar boas práticas de código (naming, comentários, estilo)
- Fazer debugging sistemático de erros
- Usar ChatGPT e Claude de forma estratégica

---

16 Revisão Rápida do Dia 1

Antes de começar, vamos relembrar os conceitos fundamentais:

```r
# Tipos de dados
numero <- 42
texto <- "Hello, R!"
logico <- TRUE
vetor <- c(1, 2, 3, 4, 5)

# Data frame
dados <- data.frame(
  nome = c("Ana", "Bruno", "Carla"),
  idade = c(25, 30, 28),
  nota = c(8.5, 7.0, 9.0)
)

# Indexação
dados$nome
```

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

```r
dados[dados$idade > 26, ]
```

```
#>    nome idade nota
#> 2 Bruno    30    7
#> 3 Carla    28    9
```

---

17 Parte 1: Operadores e Condicionais (19h00 - 20h30)

17.1 1.1 Operadores Relacionais

Operadores relacionais **comparam valores** e retornam TRUE ou FALSE.

```
# Igualdade
5 == 5    # TRUE
```

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

```
5 == 3    # FALSE
```

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

```
# Diferente
5 != 3    # TRUE
```

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

```
5 != 5    # FALSE
```

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

```
# Maior/Menor
10 > 5    # TRUE
```

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

```
10 < 5    # FALSE
```

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

```
# Maior ou igual / Menor ou igual
10 >= 10  # TRUE
```

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

```
10 <= 5   # FALSE
```

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

```
# Comparação de strings (case-sensitive)
"R" == "R"       # TRUE
```

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

```
"R" == "r"       # FALSE
```

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

```
"R" == "Python"  # FALSE
```

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

17.1.1 Aplicações Práticas

```r
# Verificar aprovação
nota <- 7.5
nota >= 7  # TRUE - Aprovado!
```

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

```r
# Verificar maioridade
idade <- 17
idade >= 18  # FALSE - Menor de idade
```

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

```r
# Comparar preços
preco_atual <- 150
preco_anterior <- 200
preco_atual < preco_anterior  # TRUE - Está mais barato!
```

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

```r
# Vetorizado
notas <- c(8.5, 6.0, 9.0, 5.5, 7.5)
notas >= 7  # TRUE FALSE TRUE FALSE TRUE
```

```
#> [1]  TRUE FALSE  TRUE FALSE  TRUE
```

17.2 1.2 Operadores Lógicos

Operadores lógicos **combinam condições**.

```r
# E (AND) - & ou &&
# Ambas condições devem ser TRUE
TRUE & TRUE    # TRUE
```

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

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

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

```r
FALSE & FALSE  # FALSE
```

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

```r
# OU (OR) - | ou ||
# Pelo menos uma condição deve ser TRUE
TRUE | FALSE   # TRUE
```

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

```r
FALSE | FALSE  # FALSE
```

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

```
# NÃO (NOT) - !
# Inverte o resultado
!TRUE   # FALSE
```

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

```
!FALSE  # TRUE
```

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

17.2.1 Diferença entre & e &&

```
# & é vetorizado (compara elemento por elemento)
c(TRUE, FALSE, TRUE) & c(TRUE, TRUE, FALSE)
```

```
#> [1]  TRUE FALSE FALSE
```

```
# && avalia apenas o primeiro elemento (usa com escalares)
TRUE && TRUE    # TRUE
```

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

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

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

```
# CUIDADO: && com vetores usa apenas primeiro elemento
# c(TRUE, FALSE) && c(TRUE, TRUE)  # Evite! Use & para vetores

# Regra: Use & para vetores, && para condições únicas em if
```

17.2.2 Aplicações Práticas

```
# Pode dirigir? (maior de idade E tem carteira)
idade <- 25
tem_carteira <- TRUE
idade >= 18 & tem_carteira  # TRUE
```

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

```
# Desconto especial (idoso OU estudante)
idade <- 70
estudante <- FALSE
idade >= 60 | estudante  # TRUE - Tem desconto!
```

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

```
# Aprovado em pelo menos uma prova
prova1 <- 6.0
prova2 <- 8.0
prova1 >= 7 | prova2 >= 7  # TRUE
```

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

```r
# Vetorizado
idades <- c(16, 22, 35, 17, 28)
tem_carteiras <- c(FALSE, TRUE, TRUE, FALSE, TRUE)

# Quem pode dirigir?
pode_dirigir <- idades >= 18 & tem_carteiras
pode_dirigir
```

```
#> [1] FALSE  TRUE  TRUE FALSE  TRUE
```

17.3 1.3 Ordem de Precedência

Quando múltiplos operadores são usados, há uma ordem de avaliação:

```r
# 1. Parênteses ()
# 2. NOT (!)
# 3. Relacionais (>, <, ==, etc)
# 4. AND (&)
# 5. OR (|)

# Exemplo
idade <- 25
tem_carteira <- TRUE
tem_carro <- FALSE

# SEM parênteses (pode confundir)
idade >= 18 & tem_carteira | tem_carro
```

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

```r
# COM parênteses (mais claro)
(idade >= 18 & tem_carteira) | tem_carro
```

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

```r
# Sempre use parênteses quando tiver dúvida!
```

17.4 1.4 Estruturas Condicionais: if e else

if executa código apenas se a condição for TRUE.

17.4.1 Sintaxe Básica

```r
# Estrutura simples
nota <- 8.5

if (nota >= 7) {
```

```
  print("Aprovado!")
}
```

```
#> [1] "Aprovado!"
```

```
# Com else
nota <- 5.5

if (nota >= 7) {
  print("Aprovado!")
} else {
  print("Reprovado!")
}
```

```
#> [1] "Reprovado!"
```

17.4.2 Múltiplas Condições: else if

```
nota <- 8.5

if (nota >= 9) {
  print("Excelente!")
} else if (nota >= 7) {
  print("Bom!")
} else if (nota >= 5) {
  print("Regular")
} else {
  print("Insuficiente")
}
```

```
#> [1] "Bom!"
```

17.4.3 Boas Práticas com if/else

```
# BOM: Use chaves mesmo para uma linha
if (x > 0) {
  print("Positivo")
}

# RUIM: Sem chaves (menos legível)
if (x > 0) print("Positivo")

# BOM: Indentação consistente
if (condicao1) {
  # código
  if (condicao2) {
    # código aninhado
  }
}
```

```r
# BOM: Condições complexas em variáveis
usuario_valido <- idade >= 18 & tem_cadastro
if (usuario_valido) {
  # código
}

# RUIM: Condição muito longa
if (idade >= 18 & tem_cadastro & email_verificado & termos_aceitos) {
  # código
}
```

17.5 1.5 ifelse() - Versão Vetorizada

Para aplicar condições em vetores inteiros, use `ifelse()`.

```r
# Sintaxe: ifelse(teste, se_verdadeiro, se_falso)

# Exemplo simples
x <- 5
resultado <- ifelse(x > 0, "Positivo", "Não positivo")
resultado
```

```
#> [1] "Positivo"
```

```r
# Vetorizado
notas <- c(8.5, 6.0, 9.0, 5.5, 7.5)
status <- ifelse(notas >= 7, "Aprovado", "Reprovado")
status
```

```
#> [1] "Aprovado"  "Reprovado" "Aprovado"  "Reprovado" "Aprovado"
```

```r
# Criar coluna em data frame
alunos <- data.frame(
  nome = c("Ana", "Bruno", "Carla", "Diego"),
  nota = c(8.5, 6.0, 9.0, 5.5)
)

alunos$status <- ifelse(alunos$nota >= 7, "Aprovado", "Reprovado")
alunos
```

```
#>    nome nota    status
#> 1   Ana  8.5  Aprovado
#> 2 Bruno  6.0 Reprovado
#> 3 Carla  9.0  Aprovado
#> 4 Diego  5.5 Reprovado
```

```r
# ifelse aninhado (evite! Use case_when)
alunos$conceito <- ifelse(alunos$nota >= 9, "A",
                   ifelse(alunos$nota >= 7, "B",
```

```
                  ifelse(alunos$nota >= 5, "C", "D")))
alunos
```

```
#>    nome nota    status conceito
#> 1   Ana  8.5  Aprovado        B
#> 2 Bruno  6.0 Reprovado        C
#> 3 Carla  9.0  Aprovado        A
#> 4 Diego  5.5 Reprovado        C
```

17.6 1.6 case_when() - Múltiplas Condições

`case_when()` do dplyr é mais elegante para múltiplas condições.

```
library(tidyverse)

# Sintaxe
notas <- c(9.5, 7.5, 5.5, 8.0, 6.5, 3.0)

conceito <- case_when(
  notas >= 9 ~ "A",
  notas >= 7 ~ "B",
  notas >= 5 ~ "C",
  TRUE ~ "D"  # Caso contrário (default)
)
conceito
```

```
#> [1] "A" "B" "C" "B" "C" "D"
```

```
# Em data frame
alunos <- data.frame(
  nome = c("Ana", "Bruno", "Carla", "Diego", "Elena"),
  nota = c(9.5, 7.5, 5.5, 8.0, 6.5)
)

alunos <- alunos %>%
  mutate(
    conceito = case_when(
      nota >= 9 ~ "A - Excelente",
      nota >= 7 ~ "B - Bom",
      nota >= 5 ~ "C - Regular",
      TRUE ~ "D - Insuficiente"
    )
  )

alunos
```

```
#>    nome nota      conceito
#> 1   Ana  9.5 A - Excelente
```

```
#> 2 Bruno  7.5       B - Bom
#> 3 Carla  5.5   C - Regular
#> 4 Diego  8.0       B - Bom
#> 5 Elena  6.5   C - Regular
```

```
# Múltiplas variáveis
alunos <- alunos %>%
  mutate(
    situacao = case_when(
      nota >= 7 & nota < 9 ~ "Aprovado",
      nota >= 9 ~ "Aprovado com Distinção",
      nota >= 5 ~ "Recuperação",
      TRUE ~ "Reprovado"
    )
  )

alunos
```

```
#>    nome nota      conceito               situacao
#> 1   Ana  9.5 A - Excelente Aprovado com Distinção
#> 2 Bruno  7.5       B - Bom               Aprovado
#> 3 Carla  5.5   C - Regular           Recuperação
#> 4 Diego  8.0       B - Bom               Aprovado
#> 5 Elena  6.5   C - Regular           Recuperação
```

17.7 1.7 Operador %in%

Verifica se valores estão dentro de um conjunto.

```
# Básico
"Ana" %in% c("Ana", "Bruno", "Carla")  # TRUE
```

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

```
"João" %in% c("Ana", "Bruno", "Carla")  # FALSE
```

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

```
# Vetorizado
nomes <- c("Ana", "João", "Bruno", "Maria")
nomes %in% c("Ana", "Bruno", "Carla")
```

```
#> [1]  TRUE FALSE  TRUE FALSE
```

```
# Filtrar data frame
alunos <- data.frame(
  nome = c("Ana", "Bruno", "Carla", "Diego", "Elena"),
  nota = c(9.5, 7.5, 5.5, 8.0, 6.5)
)
```

```r
# Alunos específicos
alunos[alunos$nome %in% c("Ana", "Carla"), ]
```

```
#>    nome nota
#> 1   Ana  9.5
#> 3 Carla  5.5
```

```r
# Com dplyr
alunos %>%
  filter(nome %in% c("Ana", "Carla", "Elena"))
```

```
#>    nome nota
#> 1   Ana  9.5
#> 2 Carla  5.5
#> 3 Elena  6.5
```

```r
# Negação: NÃO está em
alunos %>%
  filter(!nome %in% c("Bruno", "Diego"))
```

```
#>    nome nota
#> 1   Ana  9.5
#> 2 Carla  5.5
#> 3 Elena  6.5
```

---

18 INTERVALO (20h30 - 20h50)

Aproveite para: - Tomar água/café - Revisar os conceitos - Tirar dúvidas no grupo

---

19 Parte 2: Loops, Funções e Boas Práticas (20h50 - 22h00)

19.1 2.1 Loops: for

Loop for repete código um número determinado de vezes.

19.1.1 Sintaxe Básica

```r
# Estrutura básica
for (i in 1:5) {
  print(i)
}
```

```
#> [1] 1
#> [1] 2
#> [1] 3
#> [1] 4
```

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

```
# Com vetor
frutas <- c("Maçã", "Banana", "Laranja")
for (fruta in frutas) {
  print(paste("Eu gosto de", fruta))
}
```

```
#> [1] "Eu gosto de Maçã"
#> [1] "Eu gosto de Banana"
#> [1] "Eu gosto de Laranja"
```

```
# Índice com seq_along (mais seguro)
for (i in seq_along(frutas)) {
  print(paste(i, "-", frutas[i]))
}
```

```
#> [1] "1 - Maçã"
#> [1] "2 - Banana"
#> [1] "3 - Laranja"
```

19.1.2 Acumulando Resultados

```
# Soma
soma <- 0
for (i in 1:10) {
  soma <- soma + i
}
soma
```

```
#> [1] 55
```

```
# Criar vetor de resultados
# RUIM: crescer vetor dentro do loop
resultados <- c()
for (i in 1:5) {
  resultados <- c(resultados, i^2)
}

# BOM: pré-alocar vetor
resultados <- numeric(5)  # Vetor de zeros
for (i in 1:5) {
  resultados[i] <- i^2
}
resultados
```

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

19.1.3 Loops Aninhados

```r
# Tabuada
for (i in 1:3) {
  for (j in 1:3) {
    print(paste(i, "x", j, "=", i * j))
  }
}
```

```
#> [1] "1 x 1 = 1"
#> [1] "1 x 2 = 2"
#> [1] "1 x 3 = 3"
#> [1] "2 x 1 = 2"
#> [1] "2 x 2 = 4"
#> [1] "2 x 3 = 6"
#> [1] "3 x 1 = 3"
#> [1] "3 x 2 = 6"
#> [1] "3 x 3 = 9"
```

19.2 2.2 Vetorização vs Loops

R é vetorizado - prefira operações vetorizadas!

```r
# LENTO: Loop
x <- 1:1000
resultado <- numeric(length(x))
for (i in seq_along(x)) {
  resultado[i] <- x[i]^2
}

# RÁPIDO: Vetorizado
resultado <- x^2

# Benchmark
library(microbenchmark)

microbenchmark(
  loop = {
    resultado <- numeric(1000)
    for (i in 1:1000) resultado[i] <- i^2
  },
  vetorizado = {
    resultado <- (1:1000)^2
  },
  times = 100
)
```

```
#> Unit: microseconds
```

```
#>        expr     min      lq      mean  median       uq     max neval
#>        loop 399.340 404.301 411.38580 407.376 411.4145 474.739   100
#>  vetorizado   1.476   1.640   1.78596   1.763   1.8655   3.198   100
```

19.2.1 Quando Usar Loops?

Use loops quando: - Cada iteração depende da anterior - Operação não é vetorizável - Código fica mais legível com loop

```
# Fibonacci (cada termo depende dos anteriores)
fib <- numeric(10)
fib[1] <- 1
fib[2] <- 1

for (i in 3:10) {
  fib[i] <- fib[i-1] + fib[i-2]
}
fib
```

```
#>  [1]  1  1  2  3  5  8 13 21 34 55
```

```
# Simulação com estado
saldo <- 1000
for (mes in 1:12) {
  juros <- saldo * 0.01
  saldo <- saldo + juros
  print(paste("Mês", mes, "- Saldo:", round(saldo, 2)))
}
```

```
#> [1] "Mês 1 - Saldo: 1010"
#> [1] "Mês 2 - Saldo: 1020.1"
#> [1] "Mês 3 - Saldo: 1030.3"
#> [1] "Mês 4 - Saldo: 1040.6"
#> [1] "Mês 5 - Saldo: 1051.01"
#> [1] "Mês 6 - Saldo: 1061.52"
#> [1] "Mês 7 - Saldo: 1072.14"
#> [1] "Mês 8 - Saldo: 1082.86"
#> [1] "Mês 9 - Saldo: 1093.69"
#> [1] "Mês 10 - Saldo: 1104.62"
#> [1] "Mês 11 - Saldo: 1115.67"
#> [1] "Mês 12 - Saldo: 1126.83"
```

19.3 2.3 Loops: while e repeat

19.3.1 while

Executa **enquanto** condição for TRUE.

```
# Básico
contador <- 1
```

```r
while (contador <= 5) {
  print(contador)
  contador <- contador + 1
}
```

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

```r
# Cuidado: loop infinito!
# while (TRUE) {
#   print("Infinito!")
# }

# Uso prático: convergência
valor <- 100
iteracoes <- 0

while (valor > 1) {
  valor <- valor / 2
  iteracoes <- iteracoes + 1
}

print(paste("Convergiu em", iteracoes, "iterações"))
```

```
#> [1] "Convergiu em 7 iterações"
```

19.3.2 repeat e break

```r
# repeat com break
contador <- 1

repeat {
  print(contador)
  contador <- contador + 1

  if (contador > 5) {
    break  # Sai do loop
  }
}
```

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

```r
# next: pula iteração
for (i in 1:5) {
  if (i == 3) {
    next  # Pula o 3
  }
  print(i)
}
```

```
#> [1] 1
#> [1] 2
#> [1] 4
#> [1] 5
```

19.4 2.4 Funções Customizadas

Funções permitem **reutilizar código** e deixá-lo mais organizado.

19.4.1 Sintaxe Básica

```r
# Estrutura
# nome_funcao <- function(argumentos) {
#   corpo da função
#   return(resultado)
# }

# Exemplo simples
saudar <- function(nome) {
  mensagem <- paste("Olá,", nome, "!")
  return(mensagem)
}

saudar("Ana")
```

```
#> [1] "Olá, Ana !"
```

```r
saudar("Bruno")
```

```
#> [1] "Olá, Bruno !"
```

```r
# Return implícito (última linha)
dobro <- function(x) {
  x * 2
}

dobro(5)
```

```
#> [1] 10
```

19.4.2 Argumentos

```
# Múltiplos argumentos
calcular_imc <- function(peso, altura) {
  imc <- peso / altura^2
  return(imc)
}

calcular_imc(70, 1.75)
```

```
#> [1] 22.85714
```

```
calcular_imc(85, 1.80)
```

```
#> [1] 26.23457
```

```
# Argumentos opcionais (valores padrão)
saudar <- function(nome, saudacao = "Olá") {
  mensagem <- paste(saudacao, nome, "!")
  return(mensagem)
}

saudar("Ana")  # Usa padrão
```

```
#> [1] "Olá Ana !"
```

```
saudar("Bruno", "Bom dia")  # Sobrescreve
```

```
#> [1] "Bom dia Bruno !"
```

```
# Argumentos nomeados
saudar(saudacao = "Boa noite", nome = "Carla")
```

```
#> [1] "Boa noite Carla !"
```

19.4.3 Validação de Argumentos

```
# Boa prática: validar entrada
calcular_imc <- function(peso, altura) {
  # Validações
  if (!is.numeric(peso) | !is.numeric(altura)) {
    stop("Peso e altura devem ser numéricos!")
  }

  if (peso <= 0 | altura <= 0) {
    stop("Peso e altura devem ser positivos!")
  }

  # Cálculo
  imc <- peso / altura^2
  return(imc)
```

```
}

calcular_imc(70, 1.75)  # OK
```

```
#> [1] 22.85714
```

```
# calcular_imc(-70, 1.75)  # Erro!
# calcular_imc("70", 1.75)  # Erro!
```

19.4.4 Retornando Múltiplos Valores

```
# Use lista para retornar múltiplos valores
calcular_imc_completo <- function(peso, altura) {
  imc <- peso / altura^2

  # Classificação
  classificacao <- case_when(
    imc < 18.5 ~ "Abaixo do peso",
    imc < 25 ~ "Peso normal",
    imc < 30 ~ "Sobrepeso",
    TRUE ~ "Obesidade"
  )

  # Retornar lista
  resultado <- list(
    imc = round(imc, 2),
    classificacao = classificacao,
    peso = peso,
    altura = altura
  )

  return(resultado)
}

# Usar
resultado <- calcular_imc_completo(70, 1.75)
resultado$imc
```

```
#> [1] 22.86
```

```
resultado$classificacao
```

```
#> [1] "Peso normal"
```

```
# Ou desempacotar
res <- calcular_imc_completo(85, 1.80)
print(paste("IMC:", res$imc, "-", res$classificacao))
```

```
#> [1] "IMC: 26.23 - Sobrepeso"
```

19.4.5 Funções Vetorizadas

```
# Função simples (não vetorizada)
classificar_nota <- function(nota) {
  if (nota >= 7) {
    return("Aprovado")
  } else {
    return("Reprovado")
  }
}

classificar_nota(8)  # OK
```

```
#> [1] "Aprovado"
```

```
# classificar_nota(c(8, 6, 9))  # Só retorna primeiro!

# Solução 1: Vetorizar manualmente
classificar_notas_vec <- function(notas) {
  sapply(notas, function(nota) {
    if (nota >= 7) "Aprovado" else "Reprovado"
  })
}

classificar_notas_vec(c(8, 6, 9))
```

```
#> [1] "Aprovado"  "Reprovado" "Aprovado"
```

```
# Solução 2: Usar ifelse dentro
classificar_notas_v2 <- function(notas) {
  ifelse(notas >= 7, "Aprovado", "Reprovado")
}

classificar_notas_v2(c(8, 6, 9))
```

```
#> [1] "Aprovado"  "Reprovado" "Aprovado"
```

19.5 2.5 Boas Práticas de Código

19.5.1 Naming Conventions

```
# R usa snake_case (recomendado)
calcular_media_ponderada <- function(x, pesos) { }
dados_processados <- read_csv("dados.csv")
numero_de_alunos <- 30

# Evite camelCase (menos comum em R)
calcularMediaPonderada <- function(x, pesos) { }
```

```r
# Evite nomes muito curtos
cm <- function(x) mean(x)  # O que significa?

# Evite nomes muito longos
funcao_para_calcular_media_ponderada_de_notas <- function(x) { }

# Nomes descritivos
# BOM
calcular_media <- function(x) { }
dados_alunos <- data.frame()

# RUIM
calc <- function(x) { }
d <- data.frame()
```

19.5.2 Comentários Eficientes

```r
# BOM: Explica o "por quê"
# Remove outliers para não distorcer a média
dados_limpos <- dados[dados$valor < quantile(dados$valor, 0.95), ]

# Agrupa por região para análise comparativa
resultado <- dados %>%
  group_by(regiao) %>%
  summarize(media = mean(valor))

# RUIM: Explica o óbvio
# Calcula a média
media <- mean(dados$valor)

# Cria um data frame
df <- data.frame(x = 1:10)

# BOM: Comentário de seção
# ============================================================================
# Análise Exploratória
# ============================================================================

# RUIM: Código comentado (delete!)
# x <- 1:10
# y <- x^2
# plot(x, y)
```

19.5.3 Estilo de Código

```r
# Espaçamento
# BOM
resultado <- (x + y) * 2
if (x > 0) {
  print("Positivo")
}

# RUIM
resultado<-(x+y)*2
if(x>0){print("Positivo")}

# Indentação (2 espaços padrão R)
# BOM
if (condicao1) {
  # código
  if (condicao2) {
    # código aninhado
  }
}

# RUIM
if (condicao1) {
# código
if (condicao2) {
# código
}
}

# Comprimento de linha (max 80 caracteres)
# BOM
dados %>%
  filter(idade > 18) %>%
  select(nome, idade) %>%
  arrange(desc(idade))

# RUIM
dados %>% filter(idade > 18) %>% select(nome, idade) %>% arrange(desc(idade))

# Pacotes styler e lintr
library(styler)
style_file("meu_script.R")

library(lintr)
lint("meu_script.R")
```

19.6 2.6 Debugging

19.6.1 Lendo Mensagens de Erro

```
# Tipos de mensagem
# 1. ERROR: código não executa
# x + y  # Error: object 'x' not found

# 2. WARNING: executa mas com aviso
log(-1)  # Warning: NaNs produced
```

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

```
# 3. MESSAGE: informativo
library(tidyverse)  # Messages sobre conflitos
```

19.6.2 Anatomia de um Erro

```
# Exemplo de erro
dados <- data.frame(x = 1:3, y = 4:6)
dados$z + 10

# Error: object 'z' not found
#
# Interpretação:
# - O que: tentou acessar coluna 'z'
# - Problema: coluna não existe
# - Solução: verificar nomes com names(dados)
```

19.6.3 Estratégias de Debugging

```
# 1. print() para checkpoints
minha_funcao <- function(x) {
  print(paste("Entrada:", x))

  resultado <- x * 2
  print(paste("Depois de multiplicar:", resultado))

  resultado <- resultado + 10
  print(paste("Depois de somar:", resultado))

  return(resultado)
}

minha_funcao(5)
```

```
#> [1] "Entrada: 5"
#> [1] "Depois de multiplicar: 10"
#> [1] "Depois de somar: 20"
```

```
#> [1] 20
```

```
# 2. str() para ver estrutura
dados <- data.frame(x = 1:3, y = 4:6)
str(dados)
```

```
#> 'data.frame':    3 obs. of  2 variables:
#>  $ x: int  1 2 3
#>  $ y: int  4 5 6
```

```
# 3. class() e typeof()
x <- 5
class(x)
```

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

```
typeof(x)
```

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

```
# 4. traceback() após erro
# funcao_com_erro <- function() {
#   outra_funcao()
# }
#
# outra_funcao <- function() {
#   stop("Erro!")
# }
#
# funcao_com_erro()
# traceback()

# 5. debug() e browser()
# debug(minha_funcao)
# minha_funcao(5)
# undebug(minha_funcao)
```

19.6.4 Erros Comuns

```
# 1. Objeto não existe
y + 2  # Error: object 'y' not found

# 2. Tipo errado
"abc" + 2  # Error: non-numeric argument

# 3. Parênteses/aspas não fechados
print("texto
# + (espera mais entrada)

# 4. Vírgulas faltando
```

```r
c(1 2 3)  # Error

# 5. Indexação fora dos limites
x <- c(1, 2, 3)
x[5]  # NA (não erro!)

# 6. Pacote não carregado
ggplot(data)  # Error: could not find function
library(ggplot2)  # Solução
```

19.7 2.7 IA como Assistente

19.7.1 ChatGPT vs Claude

Use ChatGPT para: - Debugging rápido - Código curto (< 50 linhas) - Explicações diretas - Sugestões rápidas

Use Claude para: - Revisão de código longo - Explicações conceituais profundas - Documentação completa - Análise de scripts complexos

19.7.2 Boas Práticas com IA

```r
# PROMPT RUIM
# "Como calcular IMC?"

# PROMPT BOM
# "Estou criando uma função em R para calcular IMC que recebe peso e altura.
#  A função deve:
#  1. Validar se os valores são numéricos e positivos
#  2. Calcular IMC = peso / altura^2
#  3. Retornar IMC arredondado para 2 casas decimais
#  Pode me ajudar?"

# PROMPT RUIM
# "Meu código não funciona"

# PROMPT BOM
# "Tenho este código em R que calcula média de notas:
#
# calcular_media <- function(notas) {
#   soma <- 0
#   for (i in notas) {
#     soma <- soma + i
#   }
#   media <- soma / length(nota)
#   return(media)
# }
```

```r
#
# Quando executo calcular_media(c(8, 7, 9)), recebo erro:
# 'object nota not found'. O que está errado?"

# SEMPRE:
# 1. Forneça contexto completo
# 2. Cole código exato (se possível)
# 3. Cole mensagem de erro completa
# 4. Explique o que tentou fazer
# 5. TESTE o código sugerido
# 6. ENTENDA antes de usar
```

19.7.3 Exemplo de Uso de IA

```r
# Você tem:
calcular_media <- function(notas) {
  soma <- 0
  for (i in notas) {
    soma <- soma + i
  }
  media <- soma / length(nota)  # BUG: 'nota' deveria ser 'notas'
  return(media)
}

# ChatGPT/Claude identifica:
# - Typo: 'nota' → 'notas'
# - Sugere versão vetorizada: mean(notas)
# - Explica o erro

# Você aprende e corrige:
calcular_media <- function(notas) {
  mean(notas)  # Mais simples!
}
```

---

20 Exercícios Práticos

20.1 Exercício 1: Condicionais

```r
# a) Crie função que classifica temperatura:
#    - Abaixo de 15: "Frio"
#    - 15-25: "Agradável"
#    - Acima de 25: "Quente"

classificar_temperatura <- function(temp) {
  # Seu código aqui
```

```r
}

# Teste
classificar_temperatura(10)   # "Frio"
classificar_temperatura(20)   # "Agradável"
classificar_temperatura(30)   # "Quente"

# b) Versão vetorizada
temperaturas <- c(12, 18, 28, 15, 32)
# Classifique todas de uma vez
```

20.2 Exercício 2: Loops

```r
# a) Calcule o fatorial de 5 usando loop
#    5! = 5 × 4 × 3 × 2 × 1 = 120


# b) Sequência de Fibonacci até o 10º termo
#    1, 1, 2, 3, 5, 8, 13, 21, 34, 55


# c) Conte quantos números de 1 a 100 são divisíveis por 7
```

20.3 Exercício 3: Funções

```r
# a) Crie função para converter Celsius em Fahrenheit
#    Fórmula: F = C × 9/5 + 32

celsius_para_fahrenheit <- function(celsius) {
  # Seu código aqui
}

# Teste
celsius_para_fahrenheit(0)    # 32
celsius_para_fahrenheit(100)  # 212

# b) Crie função que retorna estatísticas de um vetor:
#    - média, mediana, desvio padrão, mínimo, máximo

estatisticas <- function(x) {
  # Seu código aqui
  # Retorne uma lista
}

# Teste
valores <- c(23, 45, 19, 34, 28, 52, 31)
```

```r
stats <- estatisticas(valores)
stats$media
stats$mediana
```

20.4 Exercício 4: Integração

```r
# Dataset: pacientes de uma clínica
pacientes <- data.frame(
  id = 1:20,
  idade = c(34, 45, 23, 56, 41, 29, 38, 52, 31, 47,
            39, 44, 27, 58, 33, 49, 26, 55, 36, 42),
  peso = c(70, 85, 58, 92, 75, 63, 80, 88, 65, 78,
           72, 86, 60, 95, 73, 82, 62, 90, 68, 76),
  altura = c(1.68, 1.75, 1.60, 1.82, 1.70, 1.65, 1.78, 1.80, 1.63, 1.73,
             1.69, 1.76, 1.61, 1.83, 1.71, 1.77, 1.64, 1.81, 1.67, 1.74),
  fumante = c(F, T, F, T, T, F, F, T, F, T,
              F, T, F, T, F, T, F, T, F, T)
)

# a) Crie função para calcular IMC e classificar


# b) Aplique a função a todos os pacientes


# c) Quantos pacientes estão em cada categoria de IMC?


# d) Qual a média de idade dos fumantes vs não fumantes?


# e) Use IA (ChatGPT ou Claude) para otimizar seu código
```

---

21 Prática Guiada: Função Completa

Vamos criar uma função profissional para análise de dados:

```r
#' Análise Descritiva de Variável Numérica
#'
#' @param x Vetor numérico
#' @param nome_var Nome da variável (opcional)
#' @param remover_na Remove NAs? Padrão TRUE
#' @return Lista com estatísticas descritivas
analise_descritiva <- function(x, nome_var = "Variável", remover_na = TRUE) {
```

```r
  # Validação
  if (!is.numeric(x)) {
    stop("x deve ser numérico!")
  }

  # Remover NAs se necessário
  if (remover_na) {
    x <- na.omit(x)
  }

  # Verificar se há dados
  if (length(x) == 0) {
    stop("Vetor vazio após remover NAs!")
  }

  # Calcular estatísticas
  stats <- list(
    variavel = nome_var,
    n = length(x),
    n_na = sum(is.na(x)),
    media = mean(x, na.rm = TRUE),
    mediana = median(x, na.rm = TRUE),
    desvio = sd(x, na.rm = TRUE),
    minimo = min(x, na.rm = TRUE),
    maximo = max(x, na.rm = TRUE),
    q25 = quantile(x, 0.25, na.rm = TRUE),
    q75 = quantile(x, 0.75, na.rm = TRUE)
  )

  # Retornar
  class(stats) <- "analise_desc"  # Classe customizada
  return(stats)
}

# Método print customizado
print.analise_desc <- function(x) {
  cat("\n")
  cat("Análise Descritiva:", x$variavel, "\n")
  cat(strrep("=", 50), "\n")
  cat("N:", x$n, "\n")
  cat("NAs:", x$n_na, "\n")
  cat("Média:", round(x$media, 2), "\n")
  cat("Mediana:", round(x$mediana, 2), "\n")
  cat("Desvio padrão:", round(x$desvio, 2), "\n")
  cat("Mínimo:", round(x$minimo, 2), "\n")
  cat("Máximo:", round(x$maximo, 2), "\n")
  cat("Q25:", round(x$q25, 2), "\n")
```

```r
  cat("Q75:", round(x$q75, 2), "\n")
  cat(strrep("=", 50), "\n\n")
}

# Usar
dados <- c(23, 45, 19, 34, NA, 28, 52, 31)
resultado <- analise_descritiva(dados, "Idade")
print(resultado)
```

```
#>
#> Análise Descritiva: Idade
#> ==================================================
#> N: 7
#> NAs: 0
#> Média: 33.14
#> Mediana: 31
#> Desvio padrão: 11.77
#> Mínimo: 19
#> Máximo: 52
#> Q25: 25.5
#> Q75: 39.5
#> ==================================================
```

---

22 Commit no GitHub

Vamos versionar nosso progresso:

```r
# Via Terminal
git add .
git commit -m "Dia 2: adiciona lógica, funções e boas práticas"
git push
```

Ou via RStudio: 1. Aba Git 2. Stage arquivos 3. Commit 4. Push

---

23 Para Casa

1. **Refazer** todos os exercícios sem consultar
2. **Criar** 3 funções úteis para sua área de trabalho
3. **Praticar** debugging com código propositalmente errado
4. **Usar** ChatGPT/Claude para explicar conceitos difíceis
5. **Ler** capítulos 4-5 do R for Data Science

---

24 Recursos Adicionais

24.1 Documentação

- Advanced R - Functions
- R for Data Science - Control Flow
- RStudio Debugging

24.2 Prática

- Exercism - R Track
- Codewars - R

24.3 Estilo

- Tidyverse Style Guide
- Pacote `styler` para formatação automática
- Pacote `lintr` para verificação de estilo

---

25 Dúvidas Frequentes

P: Quando usar for vs apply/map?
R: Use for quando cada iteração depende da anterior. Use apply/map para operações independentes (mais rápido e elegante).

P: ifelse vs case_when?
R: ifelse para 2 opções, case_when para 3+.

P: Como escolher entre ChatGPT e Claude?
R: ChatGPT para respostas rápidas, Claude para análises profundas.

P: Meu código funciona mas é lento. O que fazer?
R: 1) Vetorize operações, 2) Use funções do tidyverse, 3) Profile com `profvis`, 4) Pergunte para IA.

---

26 Conclusão do Dia 2

Parabéns! Você completou o Dia 2 e agora sabe:

- Usar operadores lógicos e relacionais
- Criar estruturas condicionais eficientes
- Entender quando usar loops
- Criar funções customizadas
- Aplicar boas práticas de código
- Fazer debugging sistemático
- Usar IA de forma estratégica

Amanhã: Manipulação de dados com tidyverse!

Última atualização: 2025-10-07
Contato: junqueiravinicius@hotmail.com
Repositório: https://github.com/viniciusjunqueira/curso-r-github-ia
Lattes: http://lattes.cnpq.br/4686677580216927

27 Parte 3: Manipula o de Dados

---

28 Objetivos do Dia 3

Ao final desta aula, você será capaz de:

- Entender a filosofia tidyverse e princípios de "tidy data"
- Usar os verbos essenciais do dplyr (filter, select, mutate, etc.)
- Transformar dados entre formatos wide e long (tidyr)
- Tratar valores ausentes de forma adequada
- Usar ferramentas modernas (janitor, skimr)
- Criar pipelines complexos de manipulação
- Otimizar código com ajuda de IA (Claude)

---

29 Revisão Rápida do Dia 2

```r
# Condicionais
idade <- 25
status <- ifelse(idade >= 18, "Maior", "Menor")

# Funções
calcular_media <- function(x) {
  mean(x, na.rm = TRUE)
}

notas <- c(8, 7, 9, 6)
calcular_media(notas)
```

```
#> [1] 7.5
```

---

30 Parte 1: Tidyverse e dplyr (19h00 - 20h30)

30.1 1.1 O que é Tidyverse?

Tidyverse é uma coleção de pacotes R para ciência de dados que compartilham uma filosofia comum.

```r
# Carregar tidyverse (carrega vários pacotes de uma vez)
library(tidyverse)

# Pacotes principais carregados:
# - ggplot2: visualização
# - dplyr: manipulação de dados
```

```
# - tidyr: organização de dados
# - readr: leitura de dados
# - purrr: programação funcional
# - tibble: data frames modernos
# - stringr: manipulação de strings
# - forcats: manipulação de fatores
```

30.1.1 Filosofia Tidyverse

Princípios fundamentais:

1. **Reutilizar estruturas de dados existentes**
2. **Compor funções simples usando o pipe**
3. **Abraçar programação funcional**
4. **Projetado para humanos**

30.2 1.2 Tidy Data (Dados Arrumados)

Três regras para dados tidy:

1. Cada **variável** é uma coluna
2. Cada **observação** é uma linha
3. Cada **valor** é uma célula

```
# DADOS TIDY (arrumados)
tidy_data <- tibble(
  pais = c("Brasil", "Brasil", "Argentina", "Argentina"),
  ano = c(2020, 2021, 2020, 2021),
  pib = c(1.5, 1.6, 0.4, 0.5)
)
tidy_data
```

```
#> # A tibble: 4 x 3
#>   pais        ano   pib
#>   <chr>     <dbl> <dbl>
#> 1 Brasil     2020   1.5
#> 2 Brasil     2021   1.6
#> 3 Argentina  2020   0.4
#> 4 Argentina  2021   0.5
```

```
# DADOS UNTIDY (bagunçados - formato wide)
untidy_data <- tibble(
  pais = c("Brasil", "Argentina"),
  pib_2020 = c(1.5, 0.4),
  pib_2021 = c(1.6, 0.5)
)
untidy_data
```

```
#> # A tibble: 2 x 3
```

```
#>   pais      pib_2020 pib_2021
#>   <chr>        <dbl>    <dbl>
#> 1 Brasil         1.5      1.6
#> 2 Argentina      0.4      0.5
```

```r
# Dados tidy facilitam análise!
tidy_data %>%
  group_by(pais) %>%
  summarize(media_pib = mean(pib))
```

```
#> # A tibble: 2 x 2
#>   pais      media_pib
#>   <chr>         <dbl>
#> 1 Argentina      0.45
#> 2 Brasil         1.55
```

30.3 1.3 O Operador Pipe: %>%

O **pipe** (%>%) passa o resultado de uma função como primeiro argumento da próxima.

```r
# SEM pipe (aninhado - difícil de ler)
round(mean(c(1, 2, 3, NA), na.rm = TRUE), 2)
```

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

```r
# COM pipe (sequencial - fácil de ler)
c(1, 2, 3, NA) %>%
  mean(na.rm = TRUE) %>%
  round(2)
```

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

```r
# Equivalente a:
# x <- c(1, 2, 3, NA)
# x <- mean(x, na.rm = TRUE)
# x <- round(x, 2)

# Atalho: Ctrl + Shift + M (Windows/Linux) ou Cmd + Shift + M (Mac)
```

30.3.1 Native Pipe (|>)

R 4.1+ tem pipe nativo:

```r
# Pipe nativo |> (funciona igual)
c(1, 2, 3, NA) |>
  mean(na.rm = TRUE) |>
  round(2)
```

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

```r
# Use o que preferir! Neste curso usamos %>% (mais comum)
```

30.4 1.4 Dataset de Exemplo

Vamos criar um dataset para praticar:

```r
# Dados de alunos (fictício)
alunos <- tibble(
  id = 1:10,
  nome = c("Ana Silva", "Bruno Costa", "Carla Dias", "Diego Mendes",
           "Elena Rocha", "Felipe Santos", "Gabi Oliveira", "Hugo Alves",
           "Iris Ferreira", "João Lima"),
  idade = c(23, 25, 22, 24, 23, 26, 21, 25, 24, 22),
  curso = c("Biologia", "Economia", "Biologia", "Economia", "Medicina",
            "Medicina", "Biologia", "Economia", "Medicina", "Economia"),
  nota_p1 = c(8.5, 7.0, 9.0, 6.5, 8.0, 9.5, 7.5, 8.0, 9.0, 6.0),
  nota_p2 = c(7.5, 8.0, 8.5, 7.0, 9.0, 9.0, 8.0, 7.5, 8.5, 7.5),
  frequencia = c(95, 87, 100, 78, 92, 98, 85, 90, 96, 82),
  bolsista = c(FALSE, TRUE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE)
)

alunos
```

```
#> # A tibble: 10 x 8
#>       id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>    <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#>  1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#>  2     2 Bruno Costa      25 Economia     7       8           87 TRUE
#>  3     3 Carla Dias       22 Biologia     9       8.5        100 FALSE
#>  4     4 Diego Mendes     24 Economia     6.5     7           78 TRUE
#>  5     5 Elena Rocha      23 Medicina     8       9           92 FALSE
#>  6     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#>  7     7 Gabi Oliveira    21 Biologia     7.5     8           85 TRUE
#>  8     8 Hugo Alves       25 Economia     8       7.5         90 FALSE
#>  9     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
#> 10    10 João Lima        22 Economia     6       7.5         82 TRUE
```

30.5 1.5 filter() - Filtrar Linhas

`filter()` seleciona linhas baseado em condições.

```r
# Filtrar alunos com nota_p1 maior que 8
alunos %>%
  filter(nota_p1 > 8)
```

```
#> # A tibble: 4 x 8
```

```
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#> 2     3 Carla Dias       22 Biologia     9       8.5        100 FALSE
#> 3     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#> 4     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
```

```
# Filtrar curso de Biologia
alunos %>%
  filter(curso == "Biologia")
```

```
#> # A tibble: 3 x 8
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#> 2     3 Carla Dias       22 Biologia     9       8.5        100 FALSE
#> 3     7 Gabi Oliveira    21 Biologia     7.5     8           85 TRUE
```

```
# Múltiplas condições com &
alunos %>%
  filter(nota_p1 > 8 & frequencia >= 90)
```

```
#> # A tibble: 4 x 8
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#> 2     3 Carla Dias       22 Biologia     9       8.5        100 FALSE
#> 3     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#> 4     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
```

```
# OU com |
alunos %>%
  filter(nota_p1 > 9 | nota_p2 > 9)
```

```
#> # A tibble: 1 x 8
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     6 Felipe Santos    26 Medicina     9.5       9         98 FALSE
```

```
# Usando %in%
alunos %>%
  filter(curso %in% c("Biologia", "Medicina"))
```

```
#> # A tibble: 6 x 8
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#> 2     3 Carla Dias       22 Biologia     9       8.5        100 FALSE
#> 3     5 Elena Rocha      23 Medicina     8       9           92 FALSE
#> 4     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#> 5     7 Gabi Oliveira    21 Biologia     7.5     8           85 TRUE
```

```
#> 6     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
```

```
# Negação com !
alunos %>%
  filter(!bolsista)  # Não bolsistas
```

```
#> # A tibble: 6 x 8
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#> 2     3 Carla Dias       22 Biologia     9       8.5        100 FALSE
#> 3     5 Elena Rocha      23 Medicina     8       9           92 FALSE
#> 4     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#> 5     8 Hugo Alves       25 Economia     8       7.5         90 FALSE
#> 6     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
```

```
# Filtrar por string
alunos %>%
  filter(str_detect(nome, "Silva"))  # Nomes com "Silva"
```

```
#> # A tibble: 1 x 8
#>      id nome      idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>     <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     1 Ana Silva    23 Biologia     8.5     7.5         95 FALSE
```

30.6 1.6 select() - Selecionar Colunas

select() escolhe quais colunas manter.

```
# Selecionar colunas específicas
alunos %>%
  select(nome, curso, nota_p1)
```

```
#> # A tibble: 10 x 3
#>    nome          curso    nota_p1
#>    <chr>         <chr>      <dbl>
#>  1 Ana Silva     Biologia     8.5
#>  2 Bruno Costa   Economia     7
#>  3 Carla Dias    Biologia     9
#>  4 Diego Mendes  Economia     6.5
#>  5 Elena Rocha   Medicina     8
#>  6 Felipe Santos Medicina     9.5
#>  7 Gabi Oliveira Biologia     7.5
#>  8 Hugo Alves    Economia     8
#>  9 Iris Ferreira Medicina     9
#> 10 João Lima     Economia     6
```

```
# Remover colunas com -
alunos %>%
  select(-id, -frequencia)
```

```
#> # A tibble: 10 x 6
#>    nome          idade curso    nota_p1 nota_p2 bolsista
#>    <chr>         <dbl> <chr>      <dbl>   <dbl> <lgl>
#>  1 Ana Silva        23 Biologia     8.5     7.5 FALSE
#>  2 Bruno Costa      25 Economia     7       8   TRUE
#>  3 Carla Dias       22 Biologia     9       8.5 FALSE
#>  4 Diego Mendes     24 Economia     6.5     7   TRUE
#>  5 Elena Rocha      23 Medicina     8       9   FALSE
#>  6 Felipe Santos    26 Medicina     9.5     9   FALSE
#>  7 Gabi Oliveira    21 Biologia     7.5     8   TRUE
#>  8 Hugo Alves       25 Economia     8       7.5 FALSE
#>  9 Iris Ferreira    24 Medicina     9       8.5 FALSE
#> 10 João Lima        22 Economia     6       7.5 TRUE
```

```
# Selecionar intervalo
alunos %>%
  select(nome:curso)  # De nome até curso
```

```
#> # A tibble: 10 x 3
#>    nome          idade curso
#>    <chr>         <dbl> <chr>
#>  1 Ana Silva        23 Biologia
#>  2 Bruno Costa      25 Economia
#>  3 Carla Dias       22 Biologia
#>  4 Diego Mendes     24 Economia
#>  5 Elena Rocha      23 Medicina
#>  6 Felipe Santos    26 Medicina
#>  7 Gabi Oliveira    21 Biologia
#>  8 Hugo Alves       25 Economia
#>  9 Iris Ferreira    24 Medicina
#> 10 João Lima        22 Economia
```

```
# Funções auxiliares
alunos %>%
  select(starts_with("nota"))  # Começa com "nota"
```

```
#> # A tibble: 10 x 2
#>    nota_p1 nota_p2
#>      <dbl>   <dbl>
#>  1     8.5     7.5
#>  2     7       8
#>  3     9       8.5
#>  4     6.5     7
#>  5     8       9
#>  6     9.5     9
#>  7     7.5     8
#>  8     8       7.5
#>  9     9       8.5
#> 10     6       7.5
```

```
alunos %>%
  select(ends_with("a"))  # Termina com "a"
```

```
#> # A tibble: 10 x 2
#>    frequencia bolsista
#>         <dbl> <lgl>
#>  1         95 FALSE
#>  2         87 TRUE
#>  3        100 FALSE
#>  4         78 TRUE
#>  5         92 FALSE
#>  6         98 FALSE
#>  7         85 TRUE
#>  8         90 FALSE
#>  9         96 FALSE
#> 10         82 TRUE
```

```
alunos %>%
  select(contains("curso"))  # Contém "curso"
```

```
#> # A tibble: 10 x 1
#>    curso
#>    <chr>
#>  1 Biologia
#>  2 Economia
#>  3 Biologia
#>  4 Economia
#>  5 Medicina
#>  6 Medicina
#>  7 Biologia
#>  8 Economia
#>  9 Medicina
#> 10 Economia
```

```
# Reordenar colunas
alunos %>%
  select(nome, curso, everything())  # Nome e curso primeiro
```

```
#> # A tibble: 10 x 8
#>    nome          curso       id idade nota_p1 nota_p2 frequencia bolsista
#>    <chr>         <chr>    <int> <dbl>   <dbl>   <dbl>      <dbl> <lgl>
#>  1 Ana Silva     Biologia     1    23     8.5     7.5         95 FALSE
#>  2 Bruno Costa   Economia     2    25     7       8           87 TRUE
#>  3 Carla Dias    Biologia     3    22     9       8.5        100 FALSE
#>  4 Diego Mendes  Economia     4    24     6.5     7           78 TRUE
#>  5 Elena Rocha   Medicina     5    23     8       9           92 FALSE
#>  6 Felipe Santos Medicina     6    26     9.5     9           98 FALSE
#>  7 Gabi Oliveira Biologia     7    21     7.5     8           85 TRUE
#>  8 Hugo Alves    Economia     8    25     8       7.5         90 FALSE
```

```
#>  9 Iris Ferreira Medicina     9    24     9       8.5          96 FALSE
#> 10 João Lima     Economia    10    22     6       7.5          82 TRUE
```

```
# Renomear ao selecionar
alunos %>%
  select(estudante = nome, disciplina = curso)
```

```
#> # A tibble: 10 x 2
#>    estudante     disciplina
#>    <chr>         <chr>
#>  1 Ana Silva     Biologia
#>  2 Bruno Costa   Economia
#>  3 Carla Dias    Biologia
#>  4 Diego Mendes  Economia
#>  5 Elena Rocha   Medicina
#>  6 Felipe Santos Medicina
#>  7 Gabi Oliveira Biologia
#>  8 Hugo Alves    Economia
#>  9 Iris Ferreira Medicina
#> 10 João Lima     Economia
```

30.7 1.7 mutate() - Criar/Modificar Colunas

mutate() cria novas colunas ou modifica existentes.

```
# Criar nova coluna
alunos %>%
  mutate(media = (nota_p1 + nota_p2) / 2)
```

```
#> # A tibble: 10 x 9
#>       id nome          idade curso    nota_p1 nota_p2 frequencia bolsista media
#>    <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>    <dbl>
#>  1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE     8
#>  2     2 Bruno Costa      25 Economia     7       8           87 TRUE      7.5
#>  3     3 Carla Dias       22 Biologia     9       8.5        100 FALSE     8.75
#>  4     4 Diego Mendes     24 Economia     6.5     7           78 TRUE      6.75
#>  5     5 Elena Rocha      23 Medicina     8       9           92 FALSE     8.5
#>  6     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE     9.25
#>  7     7 Gabi Oliveira    21 Biologia     7.5     8           85 TRUE      7.75
#>  8     8 Hugo Alves       25 Economia     8       7.5         90 FALSE     7.75
#>  9     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE     8.75
#> 10    10 João Lima        22 Economia     6       7.5         82 TRUE      6.75
```

```
# Múltiplas colunas
alunos %>%
  mutate(
    media = (nota_p1 + nota_p2) / 2,
    aprovado = media >= 7,
    conceito = case_when(
```

```
      media >= 9 ~ "A",
      media >= 7 ~ "B",
      media >= 5 ~ "C",
      TRUE ~ "D"
    )
  )
```

```
#> # A tibble: 10 x 11
#>       id nome          idade curso   nota_p1 nota_p2 frequencia bolsista media aprovado conc
#>    <int> <chr>         <dbl> <chr>     <dbl>   <dbl>      <dbl> <lgl>    <dbl> <lgl>    <chr
#>  1     1 Ana Silva        23 Biolog~     8.5     7.5         95 FALSE     8    TRUE     B
#>  2     2 Bruno Costa      25 Econom~     7       8           87 TRUE      7.5  TRUE     B
#>  3     3 Carla Dias       22 Biolog~     9       8.5        100 FALSE     8.75 TRUE     B
#>  4     4 Diego Mendes     24 Econom~     6.5     7           78 TRUE      6.75 FALSE    C
#>  5     5 Elena Rocha      23 Medici~     8       9           92 FALSE     8.5  TRUE     B
#>  6     6 Felipe Santos    26 Medici~     9.5     9           98 FALSE     9.25 TRUE     A
#>  7     7 Gabi Oliveira    21 Biolog~     7.5     8           85 TRUE      7.75 TRUE     B
#>  8     8 Hugo Alves       25 Econom~     8       7.5         90 FALSE     7.75 TRUE     B
#>  9     9 Iris Ferreira    24 Medici~     9       8.5         96 FALSE     8.75 TRUE     B
#> 10    10 João Lima        22 Econom~     6       7.5         82 TRUE      6.75 FALSE    C
```

```
# Modificar coluna existente
alunos %>%
  mutate(idade = idade + 1)  # Aniversário!
```

```
#> # A tibble: 10 x 8
#>       id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>    <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#>  1     1 Ana Silva        24 Biologia     8.5     7.5         95 FALSE
#>  2     2 Bruno Costa      26 Economia     7       8           87 TRUE
#>  3     3 Carla Dias       23 Biologia     9       8.5        100 FALSE
#>  4     4 Diego Mendes     25 Economia     6.5     7           78 TRUE
#>  5     5 Elena Rocha      24 Medicina     8       9           92 FALSE
#>  6     6 Felipe Santos    27 Medicina     9.5     9           98 FALSE
#>  7     7 Gabi Oliveira    22 Biologia     7.5     8           85 TRUE
#>  8     8 Hugo Alves       26 Economia     8       7.5         90 FALSE
#>  9     9 Iris Ferreira    25 Medicina     9       8.5         96 FALSE
#> 10    10 João Lima        23 Economia     6       7.5         82 TRUE
```

```
# Usar coluna recém-criada
alunos %>%
  mutate(
    media = (nota_p1 + nota_p2) / 2,
    media_ajustada = media * 1.1,  # Usa 'media' criada acima
    passou = media_ajustada >= 7
  )
```

```
#> # A tibble: 10 x 11
#>       id nome        idade curso nota_p1 nota_p2 frequencia bolsista media media_ajustada pa
```

```
#>    <int> <chr>       <dbl> <chr>   <dbl>   <dbl>      <dbl> <lgl>    <dbl>          <dbl> <l
#>  1     1 Ana Silva      23 Biol~     8.5     7.5         95 FALSE     8               8.8  TR
#>  2     2 Bruno Costa    25 Econ~     7       8           87 TRUE      7.5             8.25 TR
#>  3     3 Carla Dias     22 Biol~     9       8.5        100 FALSE     8.75            9.62 TR
#>  4     4 Diego Mend~    24 Econ~     6.5     7           78 TRUE      6.75            7.43 TR
#>  5     5 Elena Rocha    23 Medi~     8       9           92 FALSE     8.5             9.35 TR
#>  6     6 Felipe San~    26 Medi~     9.5     9           98 FALSE     9.25           10.2  TR
#>  7     7 Gabi Olive~    21 Biol~     7.5     8           85 TRUE      7.75            8.52 TR
#>  8     8 Hugo Alves     25 Econ~     8       7.5         90 FALSE     7.75            8.52 TR
#>  9     9 Iris Ferre~    24 Medi~     9       8.5         96 FALSE     8.75            9.62 TR
#> 10    10 João Lima      22 Econ~     6       7.5         82 TRUE      6.75            7.43 TR
```

```r
# Operações vetorizadas
alunos %>%
  mutate(
    nota_p1_pct = nota_p1 / 10 * 100,  # Converter para percentual
    nome_upper = str_to_upper(nome),    # MAIÚSCULAS
    sobrenome = str_extract(nome, "\\w+$")  # Extrair sobrenome
  )
```

```
#> # A tibble: 10 x 11
#>       id nome   idade curso nota_p1 nota_p2 frequencia bolsista nota_p1_pct nome_upper sobre
#>    <int> <chr>  <dbl> <chr>   <dbl>   <dbl>      <dbl> <lgl>          <dbl> <chr>      <chr>
#>  1     1 Ana S~    23 Biol~     8.5     7.5         95 FALSE             85 ANA SILVA  Silva
#>  2     2 Bruno~    25 Econ~     7       8           87 TRUE              70 BRUNO COS~ Costa
#>  3     3 Carla~    22 Biol~     9       8.5        100 FALSE             90 CARLA DIAS Dias
#>  4     4 Diego~    24 Econ~     6.5     7           78 TRUE              65 DIEGO MEN~ Mende
#>  5     5 Elena~    23 Medi~     8       9           92 FALSE             80 ELENA ROC~ Rocha
#>  6     6 Felip~    26 Medi~     9.5     9           98 FALSE             95 FELIPE SA~ Santo
#>  7     7 Gabi ~    21 Biol~     7.5     8           85 TRUE              75 GABI OLIV~ Olive
#>  8     8 Hugo ~    25 Econ~     8       7.5         90 FALSE             80 HUGO ALVES Alves
#>  9     9 Iris ~    24 Medi~     9       8.5         96 FALSE             90 IRIS FERR~ Ferre
#> 10    10 João ~    22 Econ~     6       7.5         82 TRUE              60 JOÃO LIMA  Lima
```

30.8 1.8 arrange() - Ordenar Linhas

arrange() ordena linhas baseado em colunas.

```r
# Ordem crescente
alunos %>%
  arrange(nota_p1)
```

```
#> # A tibble: 10 x 8
#>       id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>    <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#>  1    10 João Lima        22 Economia     6       7.5         82 TRUE
#>  2     4 Diego Mendes     24 Economia     6.5     7           78 TRUE
#>  3     2 Bruno Costa      25 Economia     7       8           87 TRUE
```

```
#>  4     7 Gabi Oliveira    21 Biologia     7.5     8           85 TRUE
#>  5     5 Elena Rocha      23 Medicina     8       9           92 FALSE
#>  6     8 Hugo Alves       25 Economia     8       7.5         90 FALSE
#>  7     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#>  8     3 Carla Dias       22 Biologia     9       8.5        100 FALSE
#>  9     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
#> 10     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
```

```
# Ordem decrescente com desc()
alunos %>%
  arrange(desc(nota_p1))
```

```
#> # A tibble: 10 x 8
#>       id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>    <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#>  1     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#>  2     3 Carla Dias       22 Biologia     9       8.5        100 FALSE
#>  3     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
#>  4     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#>  5     5 Elena Rocha      23 Medicina     8       9           92 FALSE
#>  6     8 Hugo Alves       25 Economia     8       7.5         90 FALSE
#>  7     7 Gabi Oliveira    21 Biologia     7.5     8           85 TRUE
#>  8     2 Bruno Costa      25 Economia     7       8           87 TRUE
#>  9     4 Diego Mendes     24 Economia     6.5     7           78 TRUE
#> 10    10 João Lima        22 Economia     6       7.5         82 TRUE
```

```
# Múltiplas colunas (desempate)
alunos %>%
  arrange(curso, desc(nota_p1))
```

```
#> # A tibble: 10 x 8
#>       id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>    <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#>  1     3 Carla Dias       22 Biologia     9       8.5        100 FALSE
#>  2     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#>  3     7 Gabi Oliveira    21 Biologia     7.5     8           85 TRUE
#>  4     8 Hugo Alves       25 Economia     8       7.5         90 FALSE
#>  5     2 Bruno Costa      25 Economia     7       8           87 TRUE
#>  6     4 Diego Mendes     24 Economia     6.5     7           78 TRUE
#>  7    10 João Lima        22 Economia     6       7.5         82 TRUE
#>  8     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#>  9     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
#> 10     5 Elena Rocha      23 Medicina     8       9           92 FALSE
```

```
# Ordenar com NA
dados_com_na <- tibble(
  x = c(5, 2, NA, 1, 3),
  y = c("a", "b", "c", "d", "e")
)
```

```r
dados_com_na %>%
  arrange(x)  # NAs vão para o final por padrão
```

```
#> # A tibble: 5 x 2
#>       x y
#>   <dbl> <chr>
#> 1     1 d
#> 2     2 b
#> 3     3 e
#> 4     5 a
#> 5    NA c
```

30.9 1.9 summarize() - Resumir Dados

summarize() (ou summarise()) reduz dados a um resumo.

```r
# Resumo único
alunos %>%
  summarize(
    media_geral = mean(nota_p1),
    nota_maxima = max(nota_p1),
    nota_minima = min(nota_p1),
    desvio_padrao = sd(nota_p1),
    n_alunos = n()  # Contar linhas
  )
```

```
#> # A tibble: 1 x 5
#>   media_geral nota_maxima nota_minima desvio_padrao n_alunos
#>         <dbl>       <dbl>       <dbl>         <dbl>    <int>
#> 1         7.9         9.5           6          1.15       10
```

```r
# Múltiplas estatísticas
alunos %>%
  summarize(
    across(c(nota_p1, nota_p2),
           list(media = mean, dp = sd),
           .names = "{.col}_{.fn}")
  )
```

```
#> # A tibble: 1 x 4
#>   nota_p1_media nota_p1_dp nota_p2_media nota_p2_dp
#>           <dbl>      <dbl>         <dbl>      <dbl>
#> 1           7.9       1.15          8.05      0.685
```

```r
# Com na.rm
alunos_com_na <- alunos
alunos_com_na$nota_p1[1] <- NA

alunos_com_na %>%
```

```
  summarize(
    media_sem_na = mean(nota_p1, na.rm = TRUE),
    media_com_na = mean(nota_p1)  # Retorna NA
  )
```

```
#> # A tibble: 1 x 2
#>   media_sem_na media_com_na
#>          <dbl>        <dbl>
#> 1         7.83           NA
```

30.10 1.10 group_by() - Agrupar Dados

group_by() agrupa dados para operações por grupo.

```
# Agrupar por curso
alunos %>%
  group_by(curso) %>%
  summarize(
    n = n(),
    media_p1 = mean(nota_p1),
    media_p2 = mean(nota_p2)
  )
```

```
#> # A tibble: 3 x 4
#>   curso        n media_p1 media_p2
#>   <chr>    <int>    <dbl>    <dbl>
#> 1 Biologia     3     8.33     8
#> 2 Economia     4     6.88     7.5
#> 3 Medicina     3     8.83     8.83
```

```
# Múltiplos grupos
alunos %>%
  group_by(curso, bolsista) %>%
  summarize(
    n = n(),
    media_p1 = mean(nota_p1),
    .groups = "drop"  # Remove agrupamento após resumir
  )
```

```
#> # A tibble: 5 x 4
#>   curso    bolsista     n media_p1
#>   <chr>    <lgl>    <int>    <dbl>
#> 1 Biologia FALSE        2     8.75
#> 2 Biologia TRUE         1     7.5
#> 3 Economia FALSE        1     8
#> 4 Economia TRUE         3     6.5
#> 5 Medicina FALSE        3     8.83
```

```r
# Mutate com group_by
alunos %>%
  group_by(curso) %>%
  mutate(
    media_curso = mean(nota_p1),
    diff_da_media = nota_p1 - media_curso
  ) %>%
  ungroup()  # SEMPRE desagrupar após usar!
```

```
#> # A tibble: 10 x 10
#>       id nome          idade curso nota_p1 nota_p2 frequencia bolsista media_curso diff_da_m
#>    <int> <chr>         <dbl> <chr>   <dbl>   <dbl>      <dbl> <lgl>          <dbl>      <
#>  1     1 Ana Silva        23 Biol~     8.5     7.5         95 FALSE           8.33       0
#>  2     2 Bruno Costa      25 Econ~     7       8           87 TRUE            6.88       0
#>  3     3 Carla Dias       22 Biol~     9       8.5        100 FALSE           8.33       0
#>  4     4 Diego Mendes     24 Econ~     6.5     7           78 TRUE            6.88      -0
#>  5     5 Elena Rocha      23 Medi~     8       9           92 FALSE           8.83      -0
#>  6     6 Felipe Santos    26 Medi~     9.5     9           98 FALSE           8.83       0
#>  7     7 Gabi Oliveira    21 Biol~     7.5     8           85 TRUE            8.33      -0
#>  8     8 Hugo Alves       25 Econ~     8       7.5         90 FALSE           6.88       1
#>  9     9 Iris Ferreira    24 Medi~     9       8.5         96 FALSE           8.83       0
#> 10    10 João Lima        22 Econ~     6       7.5         82 TRUE            6.88      -0
```

```r
# count() é atalho para group_by + summarize + n()
alunos %>%
  count(curso)
```

```
#> # A tibble: 3 x 2
#>   curso        n
#>   <chr>    <int>
#> 1 Biologia     3
#> 2 Economia     4
#> 3 Medicina     3
```

```r
alunos %>%
  count(curso, bolsista)
```

```
#> # A tibble: 5 x 3
#>   curso    bolsista     n
#>   <chr>    <lgl>    <int>
#> 1 Biologia FALSE        2
#> 2 Biologia TRUE         1
#> 3 Economia FALSE        1
#> 4 Economia TRUE         3
#> 5 Medicina FALSE        3
```

---

31 INTERVALO (20h30 - 20h50)

Aproveite para: - Tomar água/café - Revisar os verbos do dplyr - Experimentar com os dados

---

32 Parte 2: Tidyr, Limpeza e Ferramentas Modernas (20h50 - 22h00)

32.1 2.1 Pipeline Completo

Combinando todos os verbos:

```
# Análise completa com pipeline
resultado <- alunos %>%
  # 1. Filtrar dados relevantes
  filter(frequencia >= 75) %>%

  # 2. Criar colunas derivadas
  mutate(
    media = (nota_p1 + nota_p2) / 2,
    status = case_when(
      media >= 7 & frequencia >= 75 ~ "Aprovado",
      media >= 5 & frequencia >= 75 ~ "Recuperação",
      TRUE ~ "Reprovado"
    )
  ) %>%

  # 3. Selecionar colunas importantes
  select(nome, curso, media, status, bolsista) %>%

  # 4. Agrupar e resumir
  group_by(curso, status) %>%
  summarize(
    n = n(),
    media_curso = mean(media),
    pct_bolsistas = mean(bolsista) * 100,
    .groups = "drop"
  ) %>%

  # 5. Ordenar resultado
  arrange(curso, desc(media_curso))

resultado
```

```
#> # A tibble: 4 x 5
#>   curso    status          n media_curso pct_bolsistas
#>   <chr>    <chr>       <int>       <dbl>         <dbl>
#> 1 Biologia Aprovado        3        8.17          33.3
```

```
#> 2 Economia Aprovado        2        7.62          50
#> 3 Economia Recuperação     2        6.75         100
#> 4 Medicina Aprovado        3        8.83           0
```

32.2 2.2 tidyr: pivot_longer() e pivot_wider()

tidyr transforma dados entre formatos wide e long.

32.2.1 pivot_longer() - Wide para Long

```
# Dados wide (uma coluna por ano)
pib_wide <- tibble(
  pais = c("Brasil", "Argentina", "Chile"),
  pib_2020 = c(1.5, 0.4, 0.3),
  pib_2021 = c(1.6, 0.5, 0.35),
  pib_2022 = c(1.7, 0.45, 0.32)
)

pib_wide
```

```
#> # A tibble: 3 x 4
#>   pais      pib_2020 pib_2021 pib_2022
#>   <chr>        <dbl>    <dbl>    <dbl>
#> 1 Brasil         1.5     1.6      1.7
#> 2 Argentina      0.4     0.5      0.45
#> 3 Chile          0.3     0.35     0.32
```

```
# Transformar para long (tidy)
pib_long <- pib_wide %>%
  pivot_longer(
    cols = starts_with("pib"),      # Colunas a transformar
    names_to = "ano",                # Nome da nova coluna de nomes
    values_to = "pib"                # Nome da nova coluna de valores
  )

pib_long
```

```
#> # A tibble: 9 x 3
#>   pais      ano        pib
#>   <chr>     <chr>    <dbl>
#> 1 Brasil    pib_2020  1.5
#> 2 Brasil    pib_2021  1.6
#> 3 Brasil    pib_2022  1.7
#> 4 Argentina pib_2020  0.4
#> 5 Argentina pib_2021  0.5
#> 6 Argentina pib_2022  0.45
#> 7 Chile     pib_2020  0.3
```

```
#> 8 Chile     pib_2021  0.35
#> 9 Chile     pib_2022  0.32
```

```
# Limpar coluna 'ano'
pib_long <- pib_long %>%
  mutate(ano = str_remove(ano, "pib_") %>% as.numeric())

pib_long
```

```
#> # A tibble: 9 x 3
#>   pais        ano   pib
#>   <chr>     <dbl> <dbl>
#> 1 Brasil     2020  1.5
#> 2 Brasil     2021  1.6
#> 3 Brasil     2022  1.7
#> 4 Argentina  2020  0.4
#> 5 Argentina  2021  0.5
#> 6 Argentina  2022  0.45
#> 7 Chile      2020  0.3
#> 8 Chile      2021  0.35
#> 9 Chile      2022  0.32
```

```
# Agora fica fácil analisar
pib_long %>%
  group_by(pais) %>%
  summarize(crescimento = last(pib) - first(pib))
```

```
#> # A tibble: 3 x 2
#>   pais      crescimento
#>   <chr>           <dbl>
#> 1 Argentina      0.05
#> 2 Brasil         0.2
#> 3 Chile          0.0200
```

32.2.2 pivot_wider() - Long para Wide

```
# Reverter para wide
pib_long %>%
  pivot_wider(
    names_from = ano,
    values_from = pib,
    names_prefix = "pib_"
  )
```

```
#> # A tibble: 3 x 4
#>   pais      pib_2020 pib_2021 pib_2022
#>   <chr>        <dbl>    <dbl>    <dbl>
#> 1 Brasil         1.5     1.6      1.7
#> 2 Argentina      0.4     0.5      0.45
#> 3 Chile          0.3     0.35     0.32
```

```r
# Exemplo: dados de vendas
vendas_long <- tibble(
  mes = rep(c("Jan", "Fev", "Mar"), each = 3),
  produto = rep(c("A", "B", "C"), 3),
  vendas = c(100, 150, 200, 120, 160, 210, 110, 155, 205)
)

vendas_long
```

```
#> # A tibble: 9 x 3
#>   mes   produto vendas
#>   <chr> <chr>    <dbl>
#> 1 Jan   A          100
#> 2 Jan   B          150
#> 3 Jan   C          200
#> 4 Fev   A          120
#> 5 Fev   B          160
#> 6 Fev   C          210
#> 7 Mar   A          110
#> 8 Mar   B          155
#> 9 Mar   C          205
```

```r
# Transformar: produtos em colunas
vendas_wide <- vendas_long %>%
  pivot_wider(
    names_from = produto,
    values_from = vendas
  )

vendas_wide
```

```
#> # A tibble: 3 x 4
#>   mes       A     B     C
#>   <chr> <dbl> <dbl> <dbl>
#> 1 Jan     100   150   200
#> 2 Fev     120   160   210
#> 3 Mar     110   155   205
```

32.3 2.3 Valores Ausentes (NA)

32.3.1 Identificar NAs

```r
# Criar dados com NA
alunos_na <- alunos
alunos_na$nota_p2[c(2, 5, 8)] <- NA
alunos_na$frequencia[c(3, 7)] <- NA
```

```
alunos_na
```

```
#> # A tibble: 10 x 8
#>       id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>    <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#>  1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#>  2     2 Bruno Costa      25 Economia     7      NA           87 TRUE
#>  3     3 Carla Dias       22 Biologia     9       8.5         NA FALSE
#>  4     4 Diego Mendes     24 Economia     6.5     7           78 TRUE
#>  5     5 Elena Rocha      23 Medicina     8      NA           92 FALSE
#>  6     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#>  7     7 Gabi Oliveira    21 Biologia     7.5     8           NA TRUE
#>  8     8 Hugo Alves       25 Economia     8      NA           90 FALSE
#>  9     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
#> 10    10 João Lima        22 Economia     6       7.5         82 TRUE
```

```
# Identificar NAs
alunos_na %>%
  summarize(
    nas_nota_p2 = sum(is.na(nota_p2)),
    nas_freq = sum(is.na(frequencia)),
    total_nas = sum(is.na(.))  # Total de NAs em todo o dataset
  )
```

```
#> # A tibble: 1 x 3
#>   nas_nota_p2 nas_freq total_nas
#>         <int>    <int>     <int>
#> 1           3        2         5
```

```
# Ver quais linhas têm NA
alunos_na %>%
  filter(is.na(nota_p2) | is.na(frequencia))
```

```
#> # A tibble: 5 x 8
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     2 Bruno Costa      25 Economia     7      NA           87 TRUE
#> 2     3 Carla Dias       22 Biologia     9       8.5         NA FALSE
#> 3     5 Elena Rocha      23 Medicina     8      NA           92 FALSE
#> 4     7 Gabi Oliveira    21 Biologia     7.5     8           NA TRUE
#> 5     8 Hugo Alves       25 Economia     8      NA           90 FALSE
```

```
# Contar NAs por coluna
alunos_na %>%
  summarize(across(everything(), ~sum(is.na(.))))
```

```
#> # A tibble: 1 x 8
#>      id  nome idade curso nota_p1 nota_p2 frequencia bolsista
#>   <int> <int> <int> <int>   <int>   <int>      <int>    <int>
#> 1     0     0     0     0       0       3          2        0
```

32.3.2 Remover NAs

```
# Remover linhas com QUALQUER NA
alunos_na %>%
  drop_na()  # tidyr
```

```
#> # A tibble: 5 x 8
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#> 2     4 Diego Mendes     24 Economia     6.5     7           78 TRUE
#> 3     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#> 4     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
#> 5    10 João Lima        22 Economia     6       7.5         82 TRUE
```

```
alunos_na %>%
  na.omit()  # base R
```

```
#> # A tibble: 5 x 8
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#> 2     4 Diego Mendes     24 Economia     6.5     7           78 TRUE
#> 3     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#> 4     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
#> 5    10 João Lima        22 Economia     6       7.5         82 TRUE
```

```
# Remover linhas com NA em colunas específicas
alunos_na %>%
  drop_na(nota_p2)
```

```
#> # A tibble: 7 x 8
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#> 2     3 Carla Dias       22 Biologia     9       8.5         NA FALSE
#> 3     4 Diego Mendes     24 Economia     6.5     7           78 TRUE
#> 4     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#> 5     7 Gabi Oliveira    21 Biologia     7.5     8           NA TRUE
#> 6     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
#> 7    10 João Lima        22 Economia     6       7.5         82 TRUE
```

```
alunos_na %>%
  drop_na(nota_p2, frequencia)
```

```
#> # A tibble: 5 x 8
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#> 2     4 Diego Mendes     24 Economia     6.5     7           78 TRUE
```

```
#> 3     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#> 4     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
#> 5    10 João Lima        22 Economia     6       7.5         82 TRUE
```

```
# Filtrar sem NAs
alunos_na %>%
  filter(!is.na(nota_p2))
```

```
#> # A tibble: 7 x 8
#>      id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>   <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#> 1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#> 2     3 Carla Dias       22 Biologia     9       8.5         NA FALSE
#> 3     4 Diego Mendes     24 Economia     6.5     7           78 TRUE
#> 4     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#> 5     7 Gabi Oliveira    21 Biologia     7.5     8           NA TRUE
#> 6     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
#> 7    10 João Lima        22 Economia     6       7.5         82 TRUE
```

32.3.3 Substituir NAs

```
# Substituir por valor fixo
alunos_na %>%
  mutate(
    nota_p2 = replace_na(nota_p2, 0),
    frequencia = replace_na(frequencia, 0)
  )
```

```
#> # A tibble: 10 x 8
#>       id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
#>    <int> <chr>         <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#>  1     1 Ana Silva        23 Biologia     8.5     7.5         95 FALSE
#>  2     2 Bruno Costa      25 Economia     7       0           87 TRUE
#>  3     3 Carla Dias       22 Biologia     9       8.5          0 FALSE
#>  4     4 Diego Mendes     24 Economia     6.5     7           78 TRUE
#>  5     5 Elena Rocha      23 Medicina     8       0           92 FALSE
#>  6     6 Felipe Santos    26 Medicina     9.5     9           98 FALSE
#>  7     7 Gabi Oliveira    21 Biologia     7.5     8            0 TRUE
#>  8     8 Hugo Alves       25 Economia     8       0           90 FALSE
#>  9     9 Iris Ferreira    24 Medicina     9       8.5         96 FALSE
#> 10    10 João Lima        22 Economia     6       7.5         82 TRUE
```

```
# Substituir por média
alunos_na %>%
  mutate(
    nota_p2 = ifelse(is.na(nota_p2), mean(nota_p2, na.rm = TRUE), nota_p2)
  )
```

```
#> # A tibble: 10 x 8
#>       id nome          idade curso    nota_p1 nota_p2 frequencia bolsista
```

```
#>    <int> <chr>          <dbl> <chr>      <dbl>   <dbl>      <dbl> <lgl>
#>  1     1 Ana Silva         23 Biologia     8.5     7.5         95 FALSE
#>  2     2 Bruno Costa       25 Economia     7       8           87 TRUE
#>  3     3 Carla Dias        22 Biologia     9       8.5         NA FALSE
#>  4     4 Diego Mendes      24 Economia     6.5     7           78 TRUE
#>  5     5 Elena Rocha       23 Medicina     8       8           92 FALSE
#>  6     6 Felipe Santos     26 Medicina     9.5     9           98 FALSE
#>  7     7 Gabi Oliveira     21 Biologia     7.5     8           NA TRUE
#>  8     8 Hugo Alves        25 Economia     8       8           90 FALSE
#>  9     9 Iris Ferreira     24 Medicina     9       8.5         96 FALSE
#> 10    10 João Lima         22 Economia     6       7.5         82 TRUE
```

```
# Substituir por valor anterior/posterior (fill)
dados_sequencia <- tibble(
  mes = 1:6,
  vendas = c(100, NA, NA, 150, NA, 200)
)

dados_sequencia %>%
  fill(vendas, .direction = "down")  # Preenche para baixo
```

```
#> # A tibble: 6 x 2
#>     mes vendas
#>   <int>  <dbl>
#> 1     1    100
#> 2     2    100
#> 3     3    100
#> 4     4    150
#> 5     5    150
#> 6     6    200
```

```
dados_sequencia %>%
  fill(vendas, .direction = "up")  # Preenche para cima
```

```
#> # A tibble: 6 x 2
#>     mes vendas
#>   <int>  <dbl>
#> 1     1    100
#> 2     2    150
#> 3     3    150
#> 4     4    150
#> 5     5    200
#> 6     6    200
```

32.4 2.4 janitor: Limpeza de Dados

janitor facilita limpeza de dados bagunçados.

```r
library(janitor)

# Dados com nomes ruins
dados_sujos <- tibble(
  `Nome Completo` = c("Ana", "Bruno"),
  `Idade (anos)` = c(25, 30),
  `Nota Final!!!` = c(8.5, 7.0),
  `E-mail` = c("ana@email.com", "bruno@email.com")
)

dados_sujos
```

```
#> # A tibble: 2 x 4
#>   `Nome Completo` `Idade (anos)` `Nota Final!!!` `E-mail`
#>   <chr>                    <dbl>           <dbl> <chr>
#> 1 Ana                         25             8.5 ana@email.com
#> 2 Bruno                       30             7   bruno@email.com
```

```r
# Limpar nomes automaticamente
dados_limpos <- dados_sujos %>%
  clean_names()

dados_limpos
```

```
#> # A tibble: 2 x 4
#>   nome_completo idade_anos nota_final e_mail
#>   <chr>              <dbl>      <dbl> <chr>
#> 1 Ana                   25        8.5 ana@email.com
#> 2 Bruno                 30        7   bruno@email.com
```

```r
names(dados_limpos)
```

```
#> [1] "nome_completo" "idade_anos"    "nota_final"    "e_mail"
```

```r
# Tabela de frequência melhorada
alunos %>%
  tabyl(curso)
```

```
#>     curso n percent
#>  Biologia 3     0.3
#>  Economia 4     0.4
#>  Medicina 3     0.3
```

```r
# Com percentuais
alunos %>%
  tabyl(curso) %>%
  adorn_pct_formatting()
```

```
#>     curso n percent
#>  Biologia 3   30.0%
#>  Economia 4   40.0%
```

```
#>  Medicina 3   30.0%
```

```r
# Tabulação cruzada
alunos %>%
  tabyl(curso, bolsista) %>%
  adorn_percentages("row") %>%
  adorn_pct_formatting() %>%
  adorn_ns()  # Adiciona contagens
```

```
#>     curso      FALSE      TRUE
#>  Biologia  66.7% (2) 33.3% (1)
#>  Economia  25.0% (1) 75.0% (3)
#>  Medicina 100.0% (3)  0.0% (0)
```

```r
# Remover linhas/colunas vazias
dados_com_vazios <- tibble(
  x = c(1, 2, NA, 4),
  y = c(NA, NA, NA, NA),
  z = c(5, 6, 7, 8)
)

dados_com_vazios %>%
  remove_empty(c("rows", "cols"))
```

```
#> # A tibble: 4 x 2
#>       x     z
#>   <dbl> <dbl>
#> 1     1     5
#> 2     2     6
#> 3    NA     7
#> 4     4     8
```

32.5 2.5 skimr: Exploração Rápida

skimr gera sumários estatísticos completos.

```r
library(skimr)

# Resumo completo do dataset
alunos %>%
  skim()
```

Table 1: Data summary

Name	Piped data
Number of rows	10
Number of columns	8
Column type frequency:	

character	2
logical	1
numeric	5
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
nome	0	1	9	13	0	10	0
curso	0	1	8	8	0	3	0

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
bolsista	0	1	0.4	FAL: 6, TRU: 4

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
id	0	1	5.50	3.03	1	3.25	5.5	7.75	10.0	
idade	0	1	23.50	1.58	21	22.25	23.5	24.75	26.0	
nota_p1	0	1	7.90	1.15	6	7.12	8.0	8.88	9.5	
nota_p2	0	1	8.05	0.69	7	7.50	8.0	8.50	9.0	
frequencia	0	1	90.30	7.23	78	85.50	91.0	95.75	100.0	

```r
# Por grupo
alunos %>%
  group_by(curso) %>%
  skim()
```

Table 5: Data summary

Name	Piped data
Number of rows	10
Number of columns	8
Column type frequency:	
character	1
logical	1
numeric	5

Group variables	curso

Variable type: character

skim_variable	curso	n_missing	complete_rate	min	max	empty	n_unique	whitespace
nome	Biologia	0	1	9	13	0	3	0
nome	Economia	0	1	9	12	0	4	0
nome	Medicina	0	1	11	13	0	3	0

Variable type: logical

skim_variable	curso	n_missing	complete_rate	mean	count
bolsista	Biologia	0	1	0.33	FAL: 2, TRU: 1
bolsista	Economia	0	1	0.75	TRU: 3, FAL: 1
bolsista	Medicina	0	1	0.00	FAL: 3

Variable type: numeric

skim_variable	curso	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
id	Biologia	0	1	3.67	3.06	1.0	2.00	3.00	5.00	7.0	
id	Economia	0	1	6.00	3.65	2.0	3.50	6.00	8.50	10.0	
id	Medicina	0	1	6.67	2.08	5.0	5.50	6.00	7.50	9.0	
idade	Biologia	0	1	22.00	1.00	21.0	21.50	22.00	22.50	23.0	
idade	Economia	0	1	24.00	1.41	22.0	23.50	24.50	25.00	25.0	
idade	Medicina	0	1	24.33	1.53	23.0	23.50	24.00	25.00	26.0	
nota_p1	Biologia	0	1	8.33	0.76	7.5	8.00	8.50	8.75	9.0	
nota_p1	Economia	0	1	6.88	0.85	6.0	6.38	6.75	7.25	8.0	
nota_p1	Medicina	0	1	8.83	0.76	8.0	8.50	9.00	9.25	9.5	
nota_p2	Biologia	0	1	8.00	0.50	7.5	7.75	8.00	8.25	8.5	
nota_p2	Economia	0	1	7.50	0.41	7.0	7.38	7.50	7.62	8.0	
nota_p2	Medicina	0	1	8.83	0.29	8.5	8.75	9.00	9.00	9.0	
frequencia	Biologia	0	1	93.33	7.64	85.0	90.00	95.00	97.50	100.0	
frequencia	Economia	0	1	84.25	5.32	78.0	81.00	84.50	87.75	90.0	
frequencia	Medicina	0	1	95.33	3.06	92.0	94.00	96.00	97.00	98.0	

```r
# Apenas variáveis numéricas
alunos %>%
  skim() %>%
  filter(skim_type == "numeric")
```

Table 9: Data summary

Name	Piped data
Number of rows	10
Number of columns	8
_____	
Column type frequency:	
numeric	5
_____	
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
id	0	1	5.50	3.03	1	3.25	5.5	7.75	10.0	
idade	0	1	23.50	1.58	21	22.25	23.5	24.75	26.0	
nota_p1	0	1	7.90	1.15	6	7.12	8.0	8.88	9.5	
nota_p2	0	1	8.05	0.69	7	7.50	8.0	8.50	9.0	
frequencia	0	1	90.30	7.23	78	85.50	91.0	95.75	100.0	

```r
# Customizar output
alunos %>%
  skim() %>%
  select(skim_variable, n_missing, numeric.mean, numeric.sd)
```

```
#> # A tibble: 8 x 4
#>   skim_variable n_missing numeric.mean numeric.sd
#>   <chr>             <int>        <dbl>      <dbl>
#> 1 nome                  0        NA         NA
#> 2 curso                 0        NA         NA
#> 3 bolsista              0        NA         NA
#> 4 id                    0         5.5        3.03
#> 5 idade                 0        23.5        1.58
#> 6 nota_p1               0         7.9        1.15
#> 7 nota_p2               0         8.05       0.685
#> 8 frequencia            0        90.3        7.23
```

32.6 2.6 Joins: Combinando Datasets

Combinar dados de múltiplas tabelas:

```r
# Tabela de alunos (simplificada)
alunos_info <- tibble(
  id = 1:5,
  nome = c("Ana", "Bruno", "Carla", "Diego", "Elena"),
```

```
  curso = c("Bio", "Eco", "Bio", "Eco", "Med")
)

# Tabela de notas
notas <- tibble(
  aluno_id = c(1, 2, 3, 4, 6),  # Note: 6 não existe em alunos_info
  disciplina = c("Mat", "Mat", "Fis", "Fis", "Qui"),
  nota = c(8.5, 7.0, 9.0, 6.5, 8.0)
)

# INNER JOIN - apenas correspondências
alunos_info %>%
  inner_join(notas, by = c("id" = "aluno_id"))
```

```
#> # A tibble: 4 x 5
#>      id nome  curso disciplina  nota
#>   <dbl> <chr> <chr> <chr>      <dbl>
#> 1     1 Ana   Bio   Mat          8.5
#> 2     2 Bruno Eco   Mat          7
#> 3     3 Carla Bio   Fis          9
#> 4     4 Diego Eco   Fis          6.5
```

```
# LEFT JOIN - mantém todos da esquerda
alunos_info %>%
  left_join(notas, by = c("id" = "aluno_id"))
```

```
#> # A tibble: 5 x 5
#>      id nome  curso disciplina  nota
#>   <dbl> <chr> <chr> <chr>      <dbl>
#> 1     1 Ana   Bio   Mat          8.5
#> 2     2 Bruno Eco   Mat          7
#> 3     3 Carla Bio   Fis          9
#> 4     4 Diego Eco   Fis          6.5
#> 5     5 Elena Med   <NA>        NA
```

```
# RIGHT JOIN - mantém todos da direita
alunos_info %>%
  right_join(notas, by = c("id" = "aluno_id"))
```

```
#> # A tibble: 5 x 5
#>      id nome  curso disciplina  nota
#>   <dbl> <chr> <chr> <chr>      <dbl>
#> 1     1 Ana   Bio   Mat          8.5
#> 2     2 Bruno Eco   Mat          7
#> 3     3 Carla Bio   Fis          9
#> 4     4 Diego Eco   Fis          6.5
#> 5     6 <NA>  <NA>  Qui          8
```

```
# FULL JOIN - mantém todos
alunos_info %>%
```

```r
  full_join(notas, by = c("id" = "aluno_id"))
```

```
#> # A tibble: 6 x 5
#>      id nome  curso disciplina  nota
#>   <dbl> <chr> <chr> <chr>      <dbl>
#> 1     1 Ana   Bio   Mat          8.5
#> 2     2 Bruno Eco   Mat          7
#> 3     3 Carla Bio   Fis          9
#> 4     4 Diego Eco   Fis          6.5
#> 5     5 Elena Med   <NA>        NA
#> 6     6 <NA>  <NA>  Qui          8
```

```r
# ANTI JOIN - linhas sem correspondência
alunos_info %>%
  anti_join(notas, by = c("id" = "aluno_id"))  # Ana e Elena não têm notas
```

```
#> # A tibble: 1 x 3
#>      id nome  curso
#>   <int> <chr> <chr>
#> 1     5 Elena Med
```

32.7 2.7 Usando Claude para Otimizar Pipelines

Claude é excelente para revisar e otimizar código tidyverse.

32.7.1 Exemplo de Prompt para Claude:

```r
# Código original (funcional mas verboso)
resultado <- alunos %>%
  filter(frequencia >= 75) %>%
  mutate(media = (nota_p1 + nota_p2) / 2) %>%
  mutate(passou = ifelse(media >= 7, "Sim", "Não")) %>%
  select(nome, media, passou) %>%
  arrange(desc(media))

# Pergunte ao Claude:
# "Este pipeline do dplyr está correto mas posso melhorá-lo?
#  Há formas mais eficientes ou elegantes de fazer o mesmo?"

# Claude pode sugerir:
# - Combinar mutates
# - Usar case_when em vez de ifelse
# - Adicionar validações
# - Melhorar legibilidade
```

---

33 Exercícios Práticos

33.1 Exercício 1: Verbos Básicos

```
# Use o dataset 'alunos' criado anteriormente

# a) Filtre alunos de Biologia com nota_p1 >= 8


# b) Selecione apenas nome, curso e nota_p1


# c) Crie coluna 'media' e outra 'aprovado' (media >= 7)


# d) Ordene por media (decrescente)


# e) Agrupe por curso e calcule média geral
```

33.2 Exercício 2: Pipeline Integrado

```
# Crie um pipeline que:
# 1. Filtra alunos com frequência >= 80
# 2. Calcula média das duas provas
# 3. Classifica: "Excelente" (>=9), "Bom" (>=7), "Regular" (<7)
# 4. Agrupa por classificação e conta quantos há em cada
# 5. Calcula percentual de cada grupo
```

33.3 Exercício 3: Tidyr

```
# Dados de temperatura (wide)
temp_wide <- tibble(
  cidade = c("São Paulo", "Rio", "BH"),
  jan = c(25, 28, 24),
  fev = c(26, 29, 25),
  mar = c(24, 27, 23)
)

# a) Transforme para formato long


# b) Calcule a temperatura média por cidade


# c) Qual cidade teve maior variação de temperatura?
```

```r
# d) Volte para formato wide
```

33.4 Exercício 4: Limpeza e Análise

```r
# Dataset com problemas
dados_problematicos <- tibble(
  `Nome Completo` = c("Ana Silva", "Bruno Costa", NA, "Carla Dias"),
  `Idade (anos)` = c(25, NA, 28, 22),
  `Renda Mensal` = c(3000, 4500, NA, 3500),
  `Estado Civil` = c("Solteira", "casado", "SOLTEIRA", "Casada")
)

# a) Limpe os nomes das colunas


# b) Padronize 'Estado Civil' (todas minúsculas, primeira letra maiúscula)


# c) Substitua NAs em 'Renda Mensal' pela mediana


# d) Remova linhas onde 'Nome Completo' é NA


# e) Crie resumo estatístico com skimr
```

33.5 Exercício 5: Análise Realista

```r
# Dados de vendas (simulado)
set.seed(123)
vendas <- tibble(
  data = rep(seq(as.Date("2024-01-01"), by = "month", length.out = 6), each = 3),
  produto = rep(c("A", "B", "C"), 6),
  vendas = round(rnorm(18, mean = 1000, sd = 200)),
  regiao = sample(c("Norte", "Sul", "Leste", "Oeste"), 18, replace = TRUE)
)

# a) Calcule vendas totais por produto


# b) Qual região teve maior média de vendas?


# c) Qual produto teve maior crescimento (primeiro vs último mês)?
```

```r
# d) Crie tabela cruzada: região x produto (vendas médias)


# e) Use Claude para otimizar seu código!
```

---

34 Prática Guiada: Análise Completa

Análise exploratória completa com tidyverse:

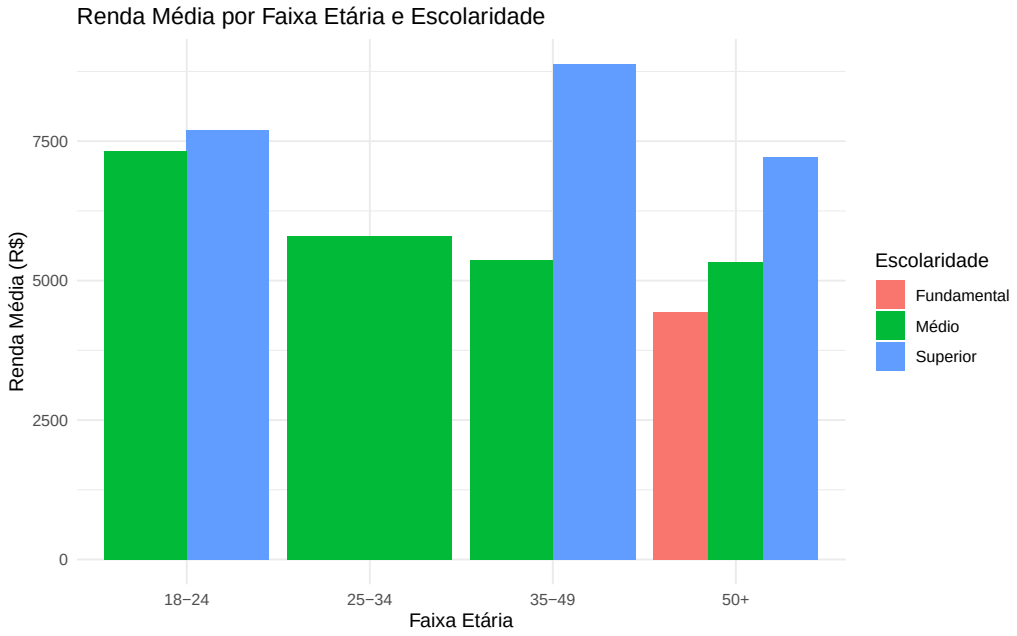
```r
# Simular dados mais complexos
set.seed(42)
n <- 100

dados_completos <- tibble(
  id = 1:n,
  idade = sample(18:65, n, replace = TRUE),
  sexo = sample(c("M", "F"), n, replace = TRUE),
  estado = sample(c("SP", "RJ", "MG", "RS"), n, replace = TRUE),
  renda = round(rnorm(n, 5000, 2000)),
  escolaridade = sample(c("Fundamental", "Médio", "Superior"), n,
                        replace = TRUE, prob = c(0.2, 0.4, 0.4)),
  satisfacao = sample(1:10, n, replace = TRUE)
) %>%
  # Criar algumas relações realistas
  mutate(
    renda = case_when(
      escolaridade == "Superior" ~ renda * 1.5,
      escolaridade == "Médio" ~ renda * 1.2,
      TRUE ~ renda
    ),
    renda = pmax(renda, 1500)  # Renda mínima
  )

# Pipeline de análise
analise <- dados_completos %>%
  # Limpeza
  filter(renda > 0, !is.na(satisfacao)) %>%

  # Engenharia de features
  mutate(
    faixa_etaria = case_when(
      idade < 25 ~ "18-24",
      idade < 35 ~ "25-34",
      idade < 50 ~ "35-49",
      TRUE ~ "50+"
    ),
```

```
    faixa_renda = case_when(
      renda < 3000 ~ "Baixa",
      renda < 7000 ~ "Média",
      TRUE ~ "Alta"
    )
  ) %>%

  # Análise por grupos
  group_by(faixa_etaria, escolaridade) %>%
  summarize(
    n = n(),
    renda_media = mean(renda),
    renda_mediana = median(renda),
    satisfacao_media = mean(satisfacao),
    pct_mulheres = mean(sexo == "F") * 100,
    .groups = "drop"
  ) %>%

  # Filtrar grupos pequenos
  filter(n >= 5) %>%

  # Ordenar
  arrange(faixa_etaria, desc(renda_media))

analise
```

```
#> # A tibble: 8 x 7
#>   faixa_etaria escolaridade     n renda_media renda_mediana satisfacao_media pct_mulheres
#>   <chr>        <chr>        <int>       <dbl>         <dbl>            <dbl>        <dbl>
#> 1 18-24        Superior         9       7706.         7480.             4.44         88.9
#> 2 18-24        Médio            5       7320.         7189.             8.4          60
#> 3 25-34        Médio            5       5791.         5156.             6.2          60
#> 4 35-49        Superior        18       8888.         8682              6.33         38.9
#> 5 35-49        Médio           13       5365.         5080.             5.85         53.8
#> 6 50+          Superior        14       7210.         7715.             5.36         42.9
#> 7 50+          Médio           17       5337.         5329.             4.18         58.8
#> 8 50+          Fundamental      7       4437          4629              6.86         28.6
```

```
# Visualização rápida do resultado
analise %>%
  ggplot(aes(x = faixa_etaria, y = renda_media, fill = escolaridade)) +
  geom_col(position = "dodge") +
  theme_minimal() +
  labs(
    title = "Renda Média por Faixa Etária e Escolaridade",
    x = "Faixa Etária",
    y = "Renda Média (R$)",
    fill = "Escolaridade"
```

```
)
```



---

35 Commit no GitHub

Versione seu progresso:

```
git add .
git commit -m "Dia 3: adiciona manipulação com tidyverse"
git push
```

---

36 Para Casa

1. **Refazer** todos os exercícios sem consultar
2. **Praticar** com seus próprios dados
3. **Ler** R for Data Science - Data Transformation
4. **Usar** Claude para revisar e otimizar seus pipelines
5. **Explorar** mais funções do dplyr: `slice()`, `distinct()`, `separate()`, `unite()`

---

37 Recursos Adicionais

37.1 Documentação

- dplyr Cheat Sheet
- tidyr Vignette
- janitor Documentation

- skimr Documentation

37.2 Prática

- TidyTuesday - Datasets semanais
- dplyr Exercises

37.3 Vídeos

- Data Wrangling with dplyr

---

38 Dúvidas Frequentes

P: Quando usar %>% vs |>?
R: Ambos funcionam igual na maioria dos casos. Use o que preferir. %>% é mais comum por enquanto.

P: Por que meu group_by não funciona?
R: Esqueceu de `ungroup()` depois? Grupos persistem até remover!

P: pivot_longer ou pivot_wider?
R: longer = wide→long (mais linhas). wider = long→wide (mais colunas).

P: Como escolher entre filter e slice?
R: filter = condição lógica. slice = por posição (linha 1, 2, 3…).

P: Meu pipeline está muito longo, é ruim?
R: Não! Pipelines longos são OK se cada passo for claro. Quebre em etapas se necessário.

---

39 Conclusão do Dia 3

Parabéns! Você completou o Dia 3 e agora domina:

- Filosofia tidyverse e tidy data
- Todos os verbos essenciais do dplyr
- Transformação de dados com tidyr
- Tratamento de valores ausentes
- Ferramentas modernas (janitor, skimr)
- Criação de pipelines complexos
- Otimização com ajuda de IA

Amanhã: Leitura/escrita de dados e visualização com ggplot2!

Última atualização: 2025-10-07
Contato: junqueiravinicius@hotmail.com
Repositório: https://github.com/viniciusjunqueira/curso-r-github-ia
Lattes: http://lattes.cnpq.br/4686677580216927

40 Parte 4: I/O e Visualiza o

---

41 Objetivos do Dia 4

Ao final desta aula, você será capaz de:

- Ler dados de diversos formatos (CSV, Excel, RDS)
- Escrever dados em diferentes formatos
- Organizar projetos com estrutura profissional
- Usar caminhos relativos com `here()`
- Entender a gramática de gráficos do ggplot2
- Criar visualizações profissionais
- Personalizar gráficos com temas e cores
- Salvar gráficos em alta qualidade
- Escolher o gráfico adequado para cada tipo de dado

---

42 Revisão Rápida do Dia 3

```
library(tidyverse)
library(here)

# Pipeline tidyverse
dados <- tibble(
  nome = c("Ana", "Bruno", "Carla"),
  nota = c(8, 7, 9),
  curso = c("Bio", "Eco", "Bio")
)

resumo <- dados %>%
  group_by(curso) %>%
  summarize(media = mean(nota))

resumo
```

```
#> # A tibble: 2 x 2
#>   curso media
#>   <chr> <dbl>
#> 1 Bio     8.5
#> 2 Eco     7
```

---

43 Parte 1: Entrada e Saída de Dados (19h00 - 20h30)

43.1 1.1 Estrutura de Projetos Profissionais

Uma boa estrutura facilita colaboração e reprodutibilidade.

43.1.1 Estrutura Recomendada

```
meu-projeto/
  meu-projeto.Rproj      # Arquivo do projeto
  README.md              # Descrição do projeto
  .gitignore             # Arquivos ignorados pelo Git

  data/                  # Dados
     raw/                # Dados originais (NUNCA modificar!)
        dados.csv
        dados.xlsx
     processed/          # Dados limpos/processados
         dados_limpos.csv

  scripts/               # Scripts R
     01_importar.R
     02_limpar.R
     03_analisar.R

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

  docs/                  # Relatórios/documentação
      relatorio.Rmd
```

43.1.2 Por que Projetos RStudio?

```
# Benefícios:
# 1. Working directory automático na raiz do projeto
# 2. Portabilidade (funciona em qualquer computador)
# 3. Integração com Git
# 4. Histórico de arquivos abertos
# 5. Configurações específicas do projeto

# Criar projeto: File > New Project
```

43.2 1.2 O Pacote here

here() cria caminhos relativos à raiz do projeto.

```r
library(here)

# Raiz do projeto
here()
```

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

```r
# Criar caminho
here("data", "raw", "dados.csv")
```

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

```r
# Vantagens sobre setwd():
# - Portável (funciona em Windows, Mac, Linux)
# - Seguro (não muda working directory global)
# - Claro (explícito onde estão os arquivos)

# RUIM (não faça):
# setwd("C:/Users/Vinicius/Documents/projeto")
# dados <- read.csv("data/dados.csv")

# BOM:
# dados <- read_csv(here("data", "raw", "dados.csv"))
```

43.3 1.3 Leitura de Arquivos CSV

CSV (Comma-Separated Values) é o formato mais comum.

43.3.1 readr::read_csv() vs base::read.csv()

```r
# readr (tidyverse) - RECOMENDADO
library(readr)

# Vantagens do readr:
# - Mais rápido (10x)
# - Retorna tibble (não data.frame)
# - Melhor tratamento de tipos
# - Progress bar para arquivos grandes
# - Encoding consistente (UTF-8)

# Criar CSV de exemplo
exemplo <- tibble(
  id = 1:5,
  nome = c("Ana", "Bruno", "Carla", "Diego", "Elena"),
  idade = c(25, 30, 28, 32, 27),
  salario = c(3000, 4500, 3500, 5000, 3800)
)
```

```r
# Salvar temporariamente
write_csv(exemplo, here("temp_exemplo.csv"))

# Ler com readr (recomendado)
dados_readr <- read_csv(here("temp_exemplo.csv"))
dados_readr
```

```
#> # A tibble: 5 x 4
#>      id nome  idade salario
#>   <dbl> <chr> <dbl>   <dbl>
#> 1     1 Ana      25    3000
#> 2     2 Bruno    30    4500
#> 3     3 Carla    28    3500
#> 4     4 Diego    32    5000
#> 5     5 Elena    27    3800
```

```r
# Ler com base R (comparação)
dados_base <- read.csv(here("temp_exemplo.csv"))
class(dados_base)  # data.frame
```

```
#> [1] "data.frame"
```

```r
class(dados_readr) # tibble
```

```
#> [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```r
# Limpar
file.remove(here("temp_exemplo.csv"))
```

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

43.3.2 Opções Importantes

```r
# Criar CSV com problemas comuns
write_lines(
  c("# Este é um comentário",
    "nome,idade,cidade",
    "Ana,25,São Paulo",
    "Bruno,,Rio de Janeiro",  # Idade ausente
    "Carla,28,",               # Cidade ausente
    "Diego,32,Belo Horizonte"),
  here("temp_problemas.csv")
)

# Ler com tratamento
dados <- read_csv(
  here("temp_problemas.csv"),
  comment = "#",           # Ignorar linhas com #
  na = c("", "NA"),        # O que considerar como NA
```

```
  col_types = cols(        # Especificar tipos
    nome = col_character(),
    idade = col_double(),
    cidade = col_character()
  )
)

dados
```

```
#> # A tibble: 4 x 3
#>   nome  idade cidade
#>   <chr> <dbl> <chr>
#> 1 Ana      25 São Paulo
#> 2 Bruno    NA Rio de Janeiro
#> 3 Carla    28 <NA>
#> 4 Diego    32 Belo Horizonte
```

```
# Ver problemas durante leitura
problems(dados)
```

```
#> # A tibble: 0 x 5
#> # i 5 variables: row <int>, col <int>, expected <chr>, actual <chr>, file <chr>
```

```
# Pular linhas
dados <- read_csv(
  here("temp_problemas.csv"),
  skip = 1  # Pular primeira linha
)

# Ler apenas algumas linhas
dados <- read_csv(
  here("temp_problemas.csv"),
  n_max = 3  # Ler apenas 3 linhas
)

# Limpar
file.remove(here("temp_problemas.csv"))
```

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

43.3.3 Outros Delimitadores

```
# TSV (tab-separated)
dados <- read_tsv("dados.tsv")

# Delimitador customizado
dados <- read_delim("dados.txt", delim = "|")

# Arquivo com vírgula como decimal (padrão brasileiro)
```

```
dados <- read_csv2("dados.csv")  # Usa ; como separador
```

43.4 1.4 Leitura de Arquivos Excel

```
library(readxl)

# Criar Excel de exemplo
library(writexl)

dados_excel <- list(
  Alunos = tibble(
    nome = c("Ana", "Bruno", "Carla"),
    nota = c(8.5, 7.0, 9.0)
  ),
  Turmas = tibble(
    turma = c("A", "B", "C"),
    n_alunos = c(30, 25, 28)
  )
)

write_xlsx(dados_excel, here("temp_exemplo.xlsx"))

# Listar sheets
excel_sheets(here("temp_exemplo.xlsx"))
```

```
#> [1] "Alunos" "Turmas"
```

```
# Ler sheet específica
alunos <- read_excel(here("temp_exemplo.xlsx"), sheet = "Alunos")
alunos
```

```
#> # A tibble: 3 x 2
#>   nome   nota
#>   <chr> <dbl>
#> 1 Ana     8.5
#> 2 Bruno   7
#> 3 Carla   9
```

```
turmas <- read_excel(here("temp_exemplo.xlsx"), sheet = 2)  # Por posição
turmas
```

```
#> # A tibble: 3 x 2
#>   turma n_alunos
#>   <chr>    <dbl>
#> 1 A           30
#> 2 B           25
#> 3 C           28
```

```
# Ler range específico
dados <- read_excel(
  here("temp_exemplo.xlsx"),
  sheet = "Alunos",
  range = "A1:B3"  # Células específicas
)

# Pular linhas
dados <- read_excel(
  here("temp_exemplo.xlsx"),
  sheet = "Alunos",
  skip = 1  # Pular cabeçalho
)

# Limpar
file.remove(here("temp_exemplo.xlsx"))
```

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

43.5 1.5 Escrita de Arquivos

43.5.1 CSV

```
# Criar dados
resultados <- tibble(
  data = Sys.Date(),
  analise = "Exploratória",
  n_observacoes = 100,
  media = 75.5
)

# Escrever CSV
write_csv(resultados, here("temp_resultados.csv"))

# Ler de volta
read_csv(here("temp_resultados.csv"))
```

```
#> # A tibble: 1 x 4
#>   data       analise      n_observacoes media
#>   <date>     <chr>                <dbl> <dbl>
#> 1 2025-10-07 Exploratória           100  75.5
```

```
# Adicionar a arquivo existente
novos_resultados <- tibble(
  data = Sys.Date() + 1,
  analise = "Confirmatória",
  n_observacoes = 150,
```

```
  media = 78.2
)

write_csv(novos_resultados, here("temp_resultados.csv"), append = TRUE)

read_csv(here("temp_resultados.csv"))
```

```
#> # A tibble: 2 x 4
#>   data       analise       n_observacoes media
#>   <date>     <chr>                 <dbl> <dbl>
#> 1 2025-10-07 Exploratória            100  75.5
#> 2 2025-10-08 Confirmatória           150  78.2
```

```
# Limpar
file.remove(here("temp_resultados.csv"))
```

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

43.5.2 Excel

```
library(writexl)

# Múltiplas sheets
dados_completos <- list(
  Resumo = tibble(total = 100, media = 75),
  Detalhado = tibble(id = 1:5, valor = c(70, 75, 80, 72, 78))
)

write_xlsx(dados_completos, here("temp_analise.xlsx"))

# Limpar
file.remove(here("temp_analise.xlsx"))
```

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

43.5.3 RDS (formato R nativo)

```
# RDS preserva TUDO: tipos, classes, atributos
modelo <- lm(mpg ~ wt, data = mtcars)

# Salvar
saveRDS(modelo, here("temp_modelo.rds"))

# Carregar
modelo_carregado <- readRDS(here("temp_modelo.rds"))
summary(modelo_carregado)
```

```
#>
#> Call:
```

```
#> lm(formula = mpg ~ wt, data = mtcars)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -4.5432 -2.3647 -0.1252  1.4096  6.8727
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
#> wt           -5.3445     0.5591  -9.559 1.29e-10 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 3.046 on 30 degrees of freedom
#> Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
#> F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

```
# Vantagens RDS:
# - Preserva estrutura completa de objetos R
# - Mais rápido que CSV
# - Compressão automática
# - Ideal para resultados intermediários

# Limpar
file.remove(here("temp_modelo.rds"))
```

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

43.6 1.6 Boas Práticas de I/O

```
# 1. SEMPRE use here() para caminhos
dados <- read_csv(here("data", "raw", "dados.csv"))

# 2. NUNCA modifique dados originais
# Leia de raw/, salve em processed/
dados_limpos <- dados %>% clean_data()
write_csv(dados_limpos, here("data", "processed", "dados_limpos.csv"))

# 3. Use nomes descritivos com data
hoje <- Sys.Date()
write_csv(resultados, here("output", paste0("resultados_", hoje, ".csv")))

# 4. Documente encoding explicitamente
dados <- read_csv(here("dados.csv"), locale = locale(encoding = "UTF-8"))

# 5. Valide dados após importar
dados <- read_csv(here("dados.csv"))
```

```r
stopifnot(nrow(dados) > 0)
stopifnot("idade" %in% names(dados))
```

---

44 INTERVALO (20h30 - 20h50)

Aproveite para: - Tomar água/café - Organizar sua estrutura de projeto - Testar leitura de seus próprios dados

---

45 Parte 2: Visualização com ggplot2 (20h50 - 22h00)

45.1 2.1 Gramática de Gráficos

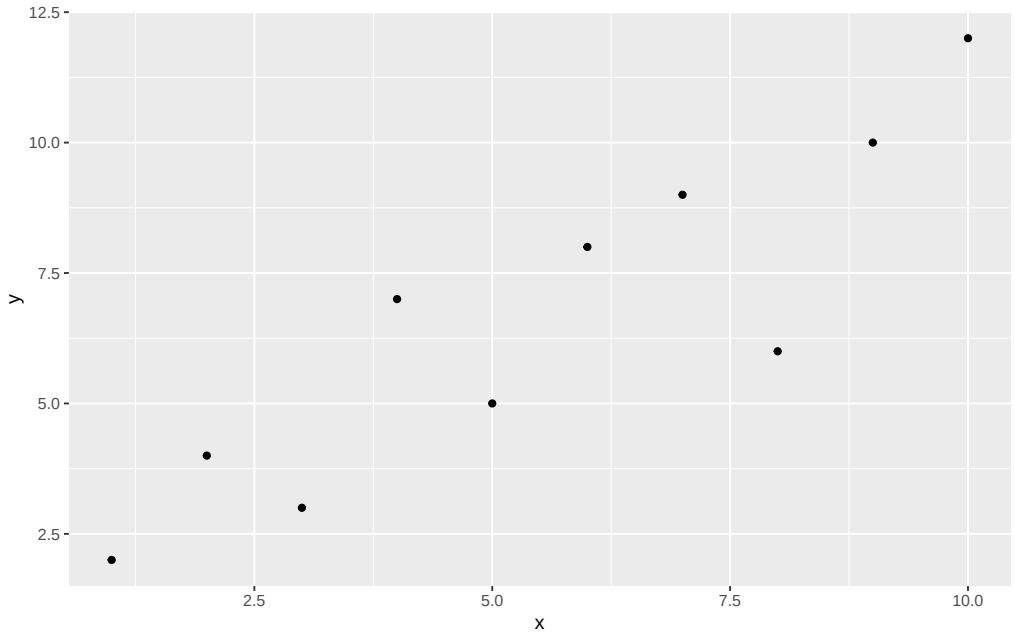
ggplot2 implementa uma "gramática de gráficos" - uma abordagem sistemática para criar visualizações.

45.1.1 Componentes Fundamentais

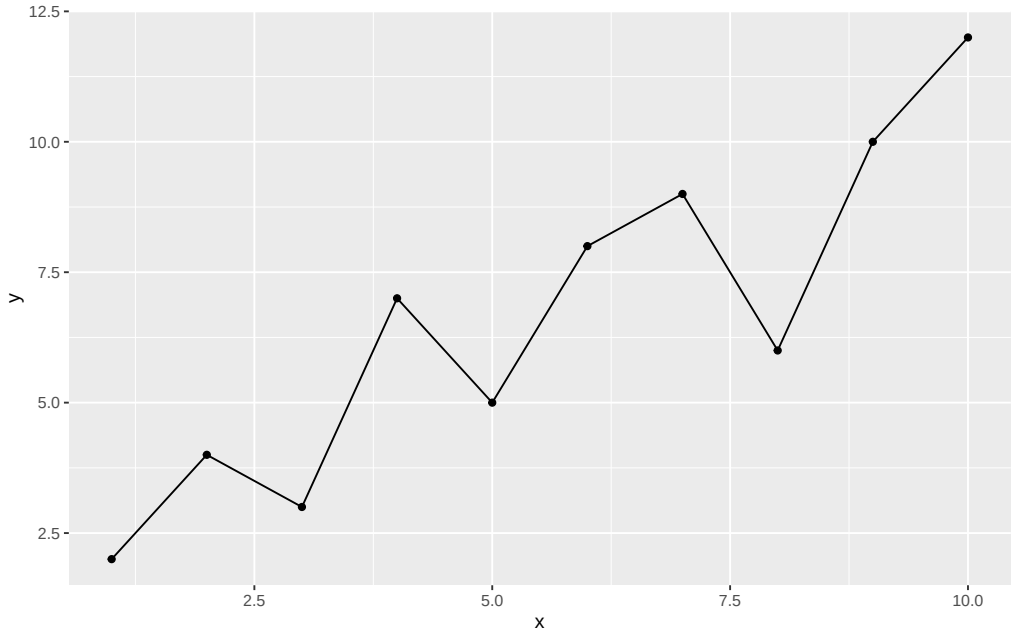
```r
# Estrutura básica
ggplot(data = dados, aes(x = variavel_x, y = variavel_y)) +
  geom_*() +        # Tipo de gráfico (pontos, linhas, barras)
  labs() +          # Títulos e labels
  theme_*()         # Tema visual

# Camadas são adicionadas com +
```

45.1.2 Primeiro Gráfico

```r
library(ggplot2)

# Dados de exemplo
dados <- tibble(
  x = 1:10,
  y = c(2, 4, 3, 7, 5, 8, 9, 6, 10, 12)
)

# Gráfico básico
ggplot(dados, aes(x = x, y = y)) +
  geom_point()
```

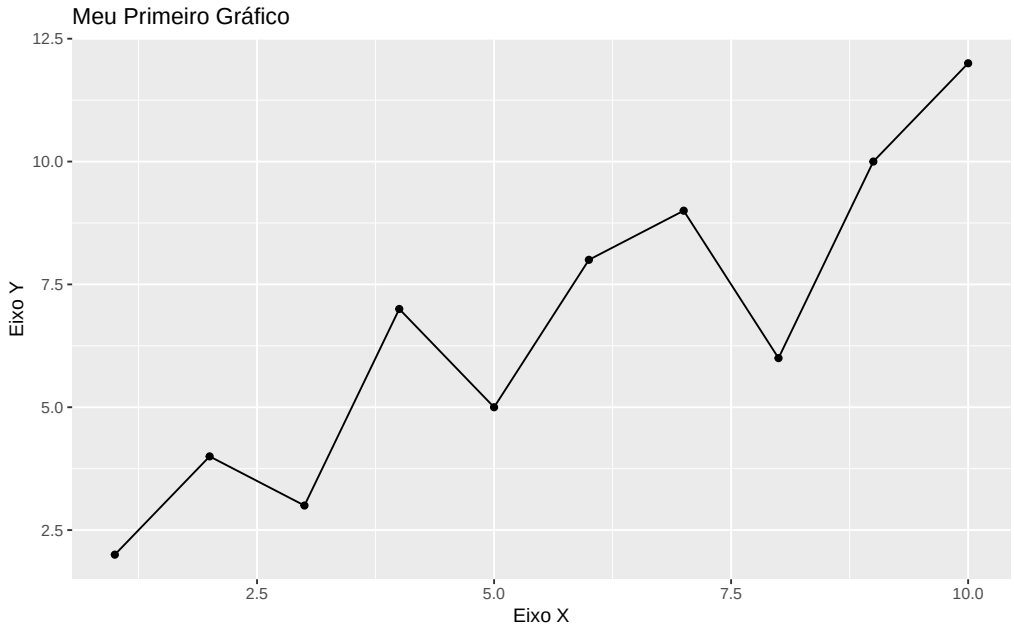



```r
# Com linha
ggplot(dados, aes(x = x, y = y)) +
  geom_point() +
  geom_line()
```




```r
# Adicionar títulos
ggplot(dados, aes(x = x, y = y)) +
  geom_point() +
  geom_line() +
  labs(
    title = "Meu Primeiro Gráfico",
    x = "Eixo X",
```

```
    y = "Eixo Y"
  )
```



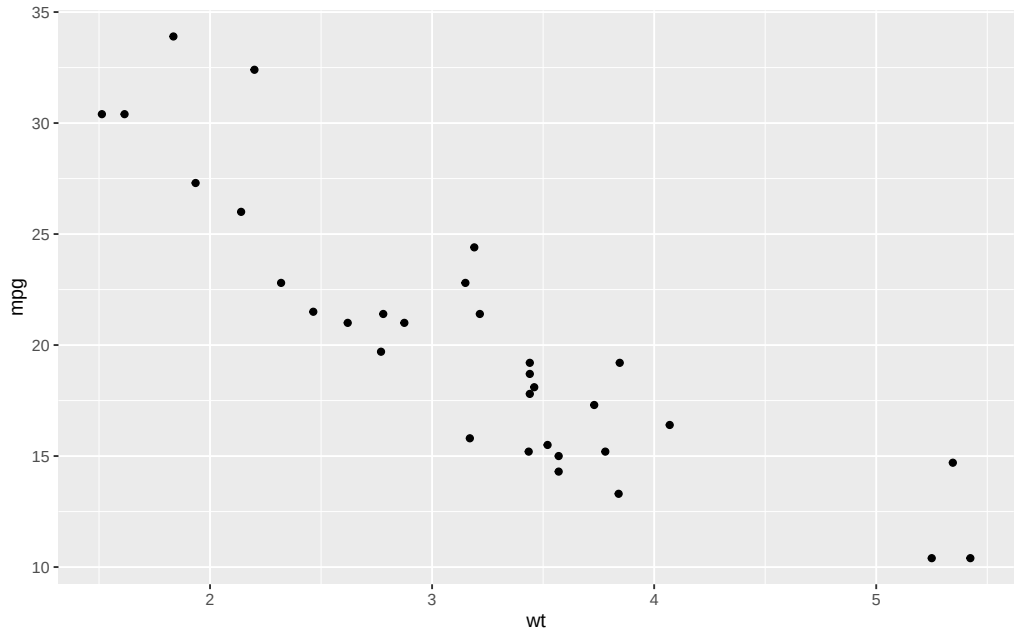
45.2 2.2 Gráfico de Dispersão

Dispersão mostra relação entre duas variáveis contínuas.

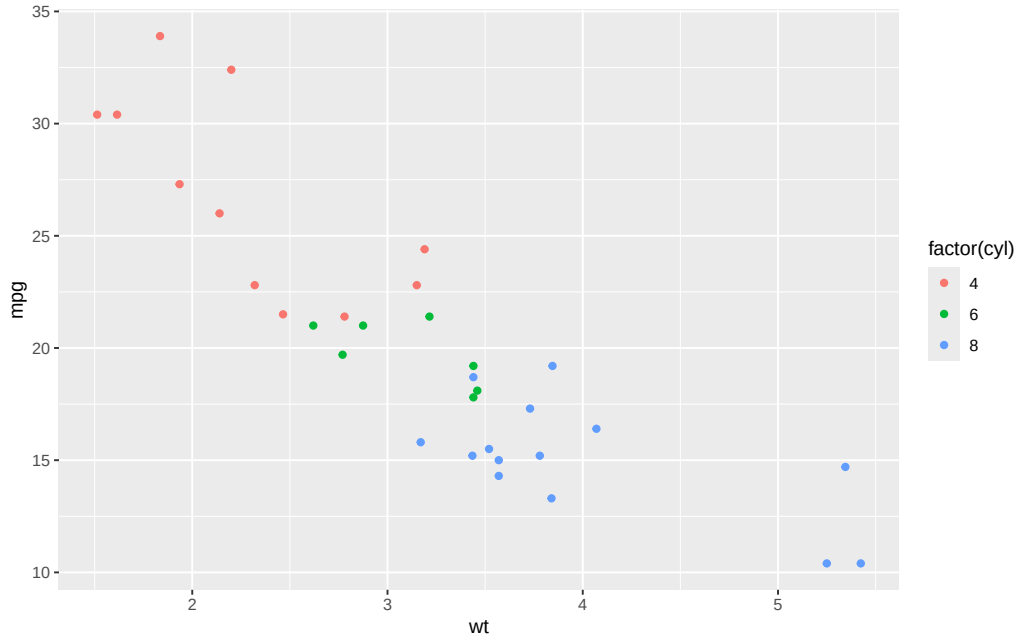
```
# Usar mtcars (dataset nativo)
glimpse(mtcars)
```

```
#> Rows: 32
#> Columns: 11
#> $ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8, 16.4, 17.3, 1
#> $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8, 8, 8, 8, 4, 4
#> $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 167.6, 167.6, 27
#> $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180, 205, 215, 23
#> $ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92, 3.07, 3.07, 3
#> $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.440, 3.440, 4.
#> $ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 18.30, 18.90, 17
#> $ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0
#> $ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1
#> $ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 4, 5
#> $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2, 2, 4, 2, 1, 2
```

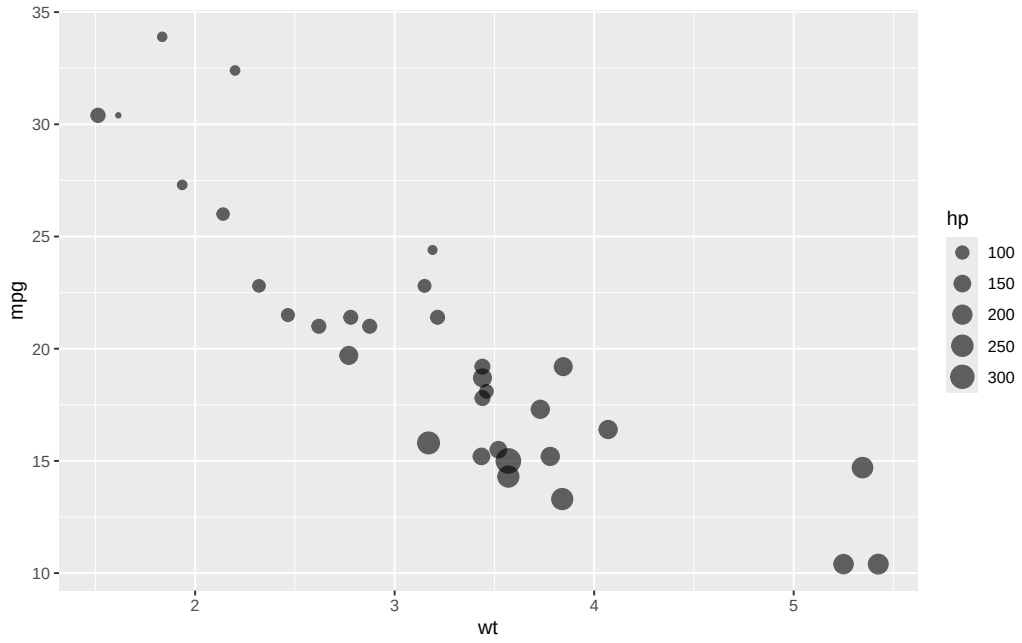
```
# Básico
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()
```



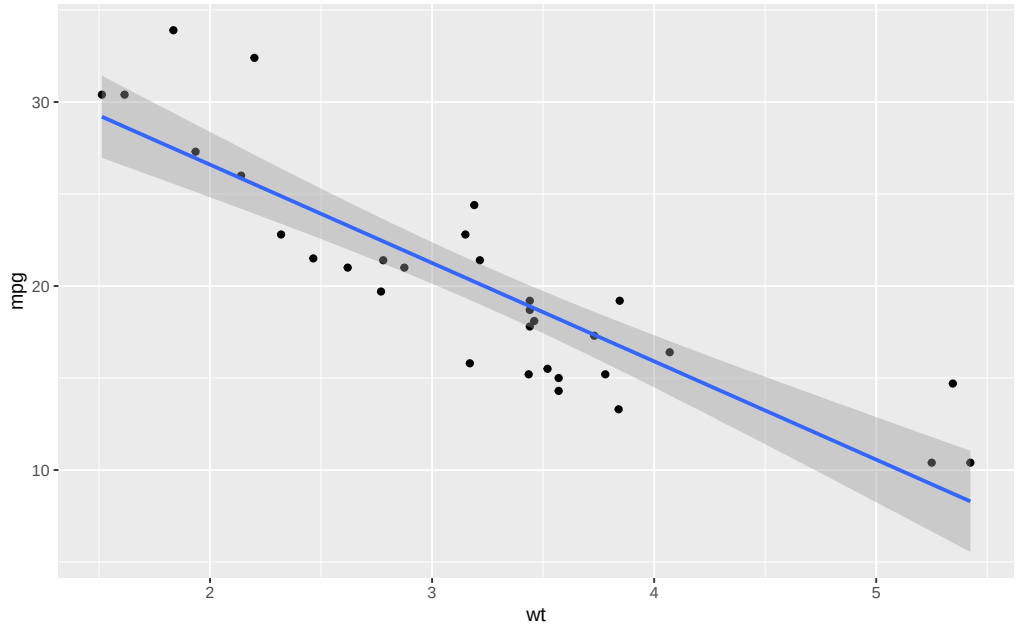

```r
# Com cor por categoria
ggplot(mtcars, aes(x = wt, y = mpg, color = factor(cyl))) +
  geom_point()
```




```r
# Com tamanho
ggplot(mtcars, aes(x = wt, y = mpg, size = hp)) +
  geom_point(alpha = 0.6)  # alpha = transparência
```

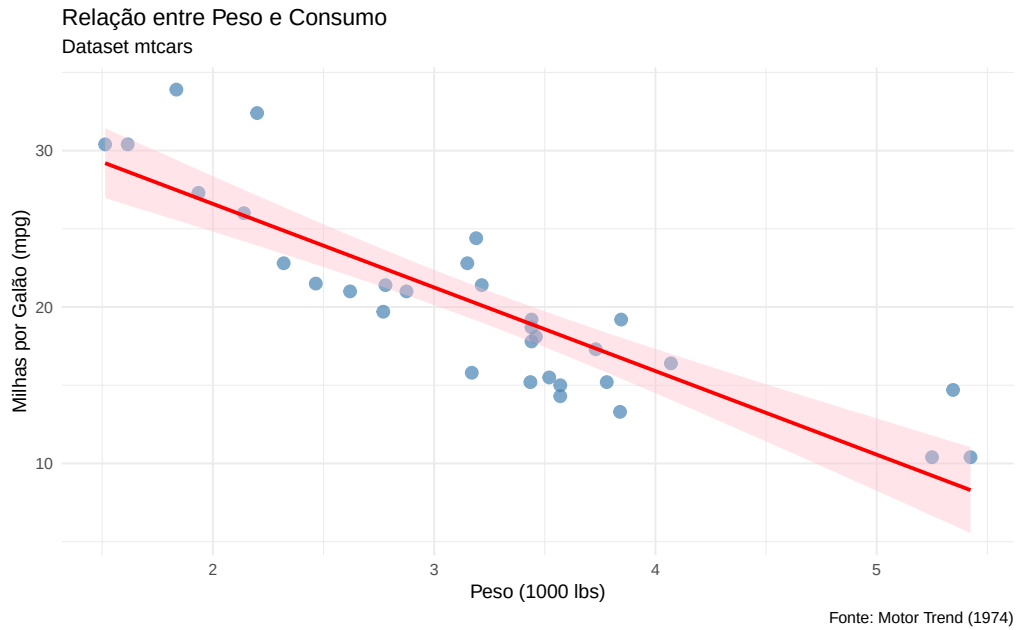



```r
# Com linha de tendência
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  geom_smooth(method = "lm", se = TRUE)  # se = intervalo de confiança
```




```r
# Customizar
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(color = "steelblue", size = 3, alpha = 0.7) +
  geom_smooth(method = "lm", color = "red", fill = "pink") +
  labs(
    title = "Relação entre Peso e Consumo",
    subtitle = "Dataset mtcars",
```

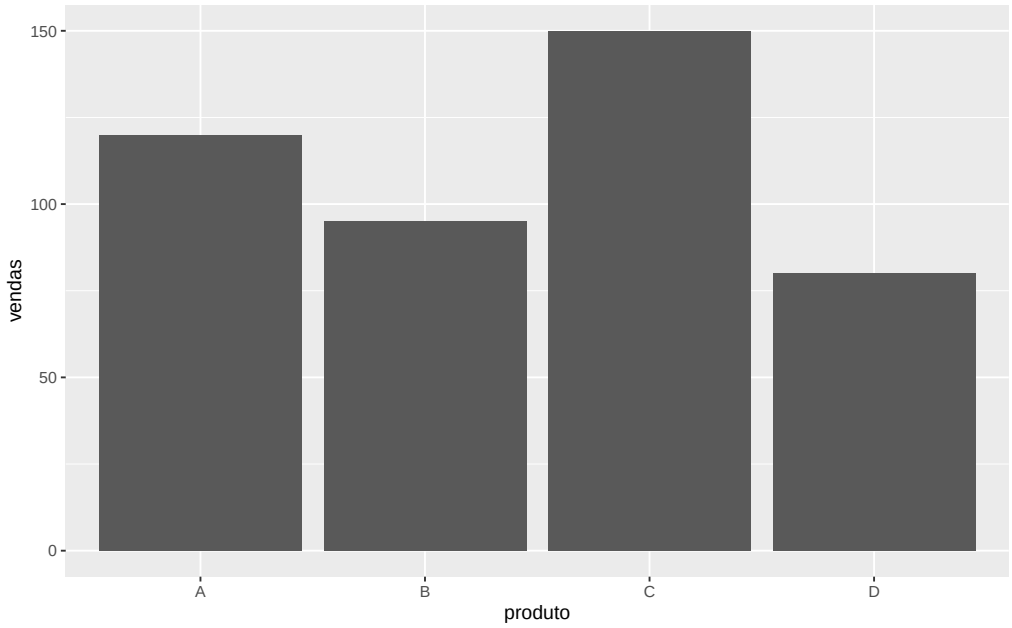
```
    x = "Peso (1000 lbs)",
    y = "Milhas por Galão (mpg)",
    caption = "Fonte: Motor Trend (1974)"
  ) +
  theme_minimal()
```



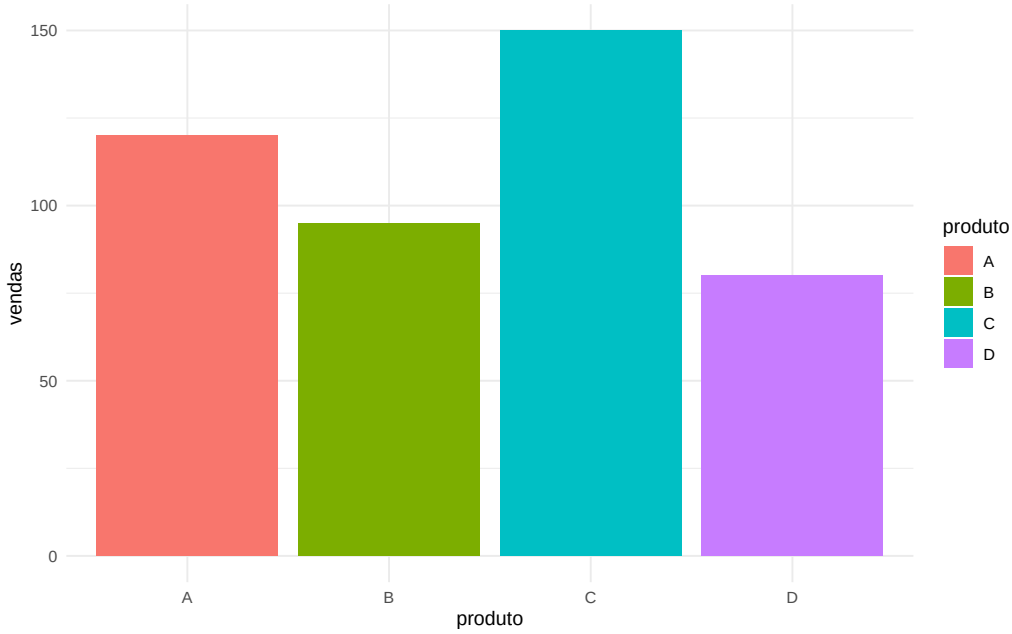
45.3 2.3 Gráfico de Barras

Barras comparam valores entre categorias.

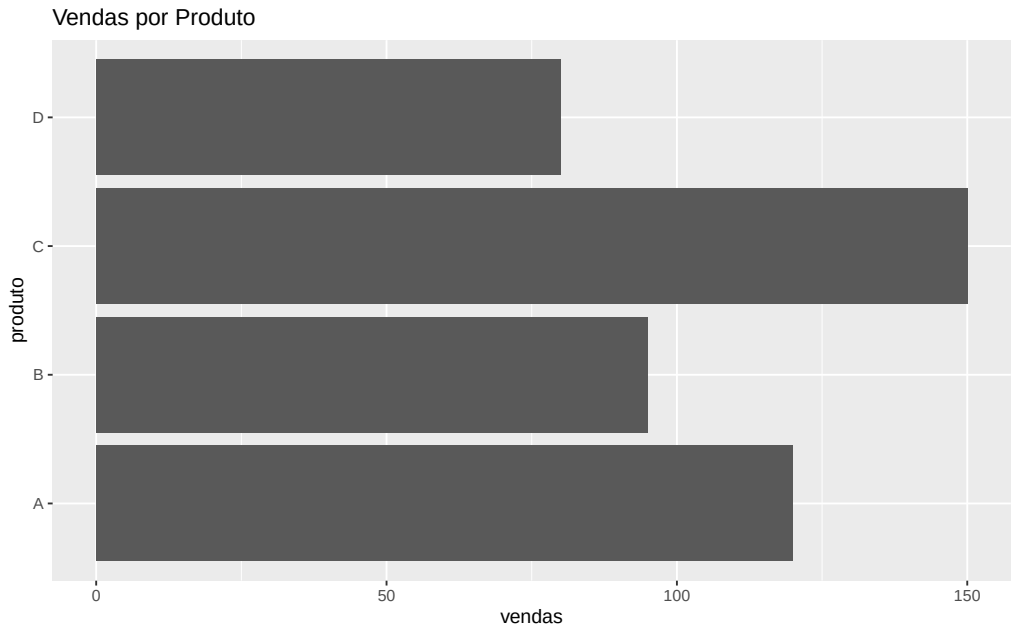
```
# Dados de exemplo
vendas <- tibble(
  produto = c("A", "B", "C", "D"),
  vendas = c(120, 95, 150, 80)
)

# geom_col() - altura especificada
ggplot(vendas, aes(x = produto, y = vendas)) +
  geom_col()
```



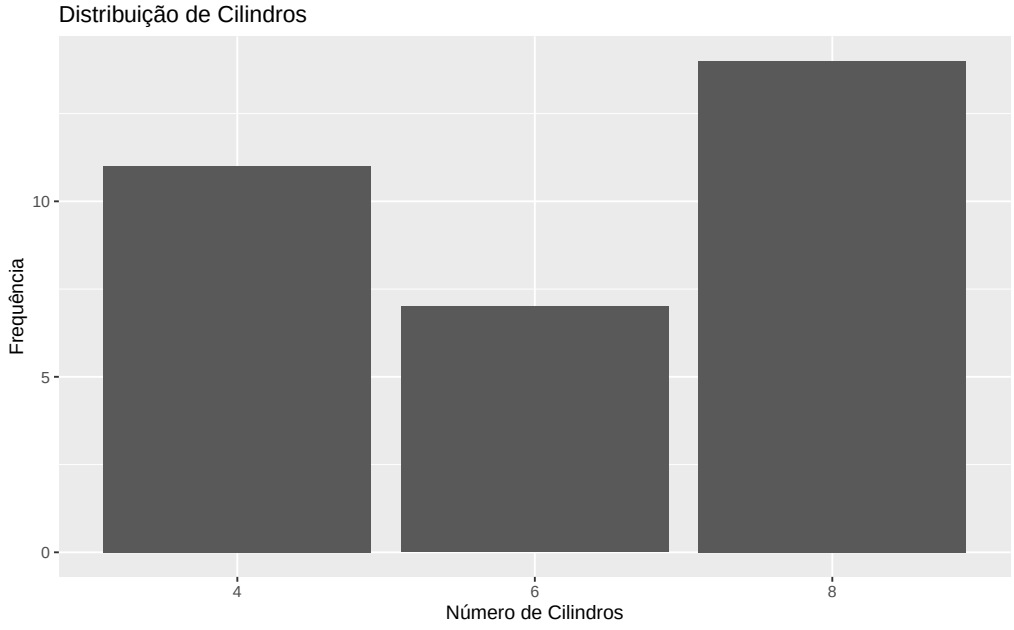

```r
# Com cores
ggplot(vendas, aes(x = produto, y = vendas, fill = produto)) +
  geom_col() +
  theme_minimal()
```




```r
# Barras horizontais
ggplot(vendas, aes(x = vendas, y = produto)) +
  geom_col() +
  labs(title = "Vendas por Produto")
```

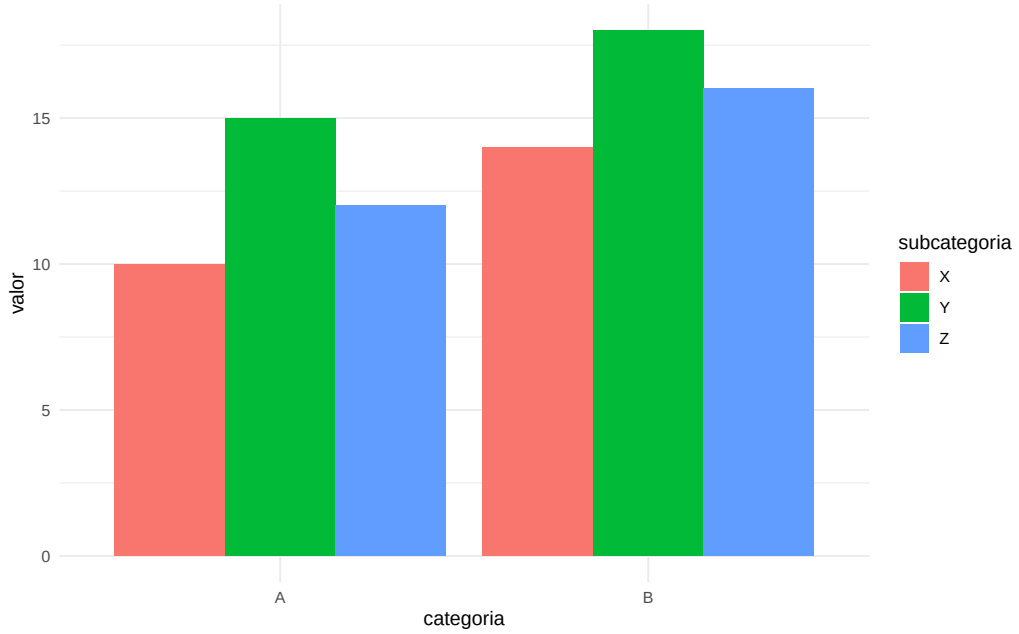



```r
# geom_bar() - conta frequências
ggplot(mtcars, aes(x = factor(cyl))) +
  geom_bar() +
  labs(
    title = "Distribuição de Cilindros",
    x = "Número de Cilindros",
    y = "Frequência"
  )
```

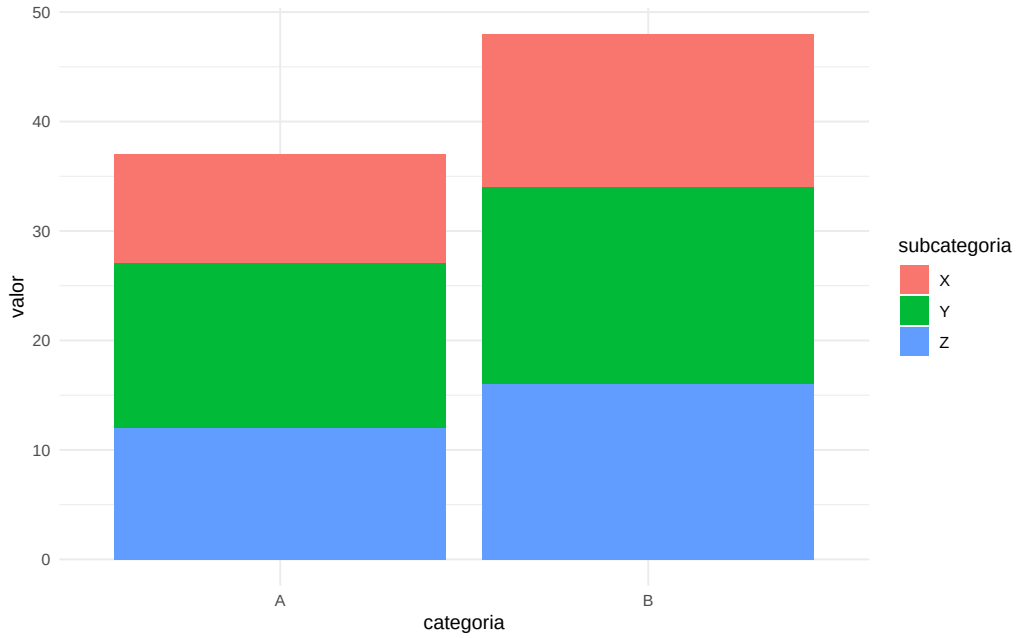



```r
# Barras agrupadas
dados_grouped <- tibble(
  categoria = rep(c("A", "B"), each = 3),
  subcategoria = rep(c("X", "Y", "Z"), 2),
```

```
  valor = c(10, 15, 12, 14, 18, 16)
)

ggplot(dados_grouped, aes(x = categoria, y = valor, fill = subcategoria)) +
  geom_col(position = "dodge") +  # lado a lado
  theme_minimal()
```



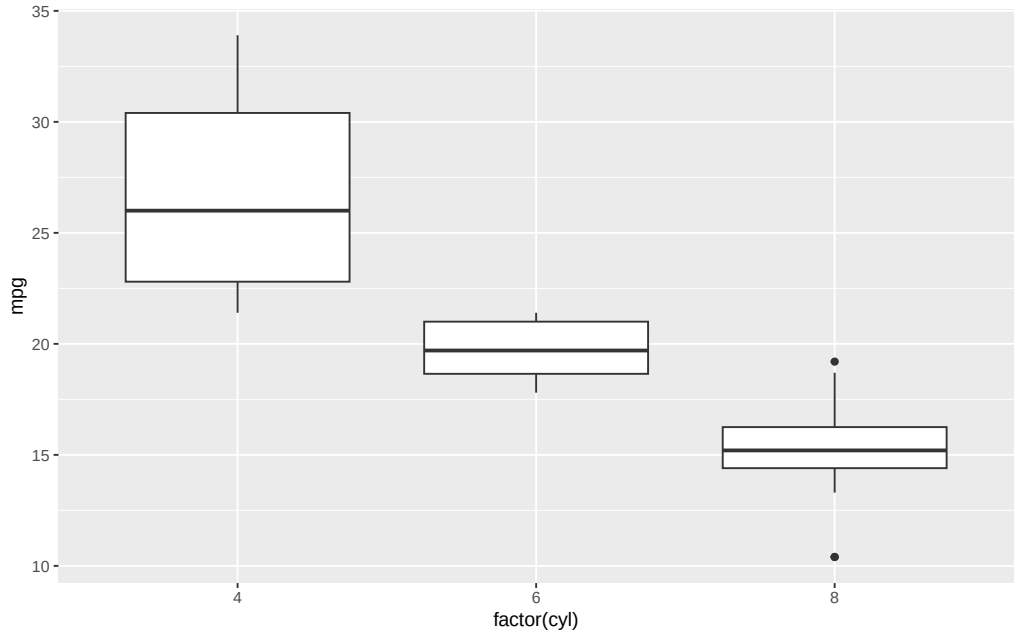

```
# Barras empilhadas
ggplot(dados_grouped, aes(x = categoria, y = valor, fill = subcategoria)) +
  geom_col(position = "stack") +
  theme_minimal()
```



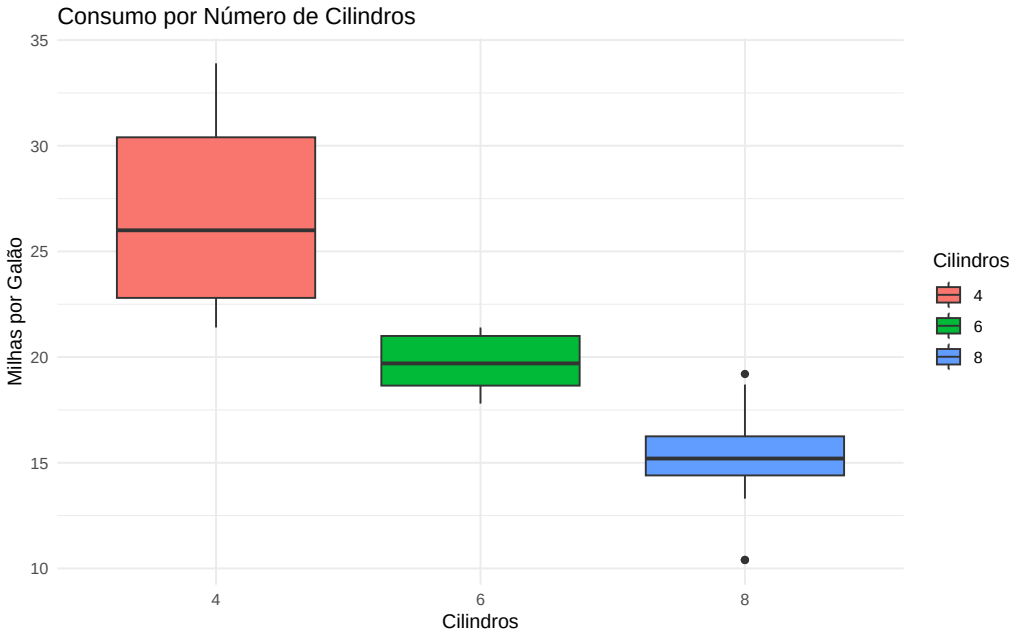
45.4 2.4 Boxplot

Boxplot mostra distribuição e identifica outliers.

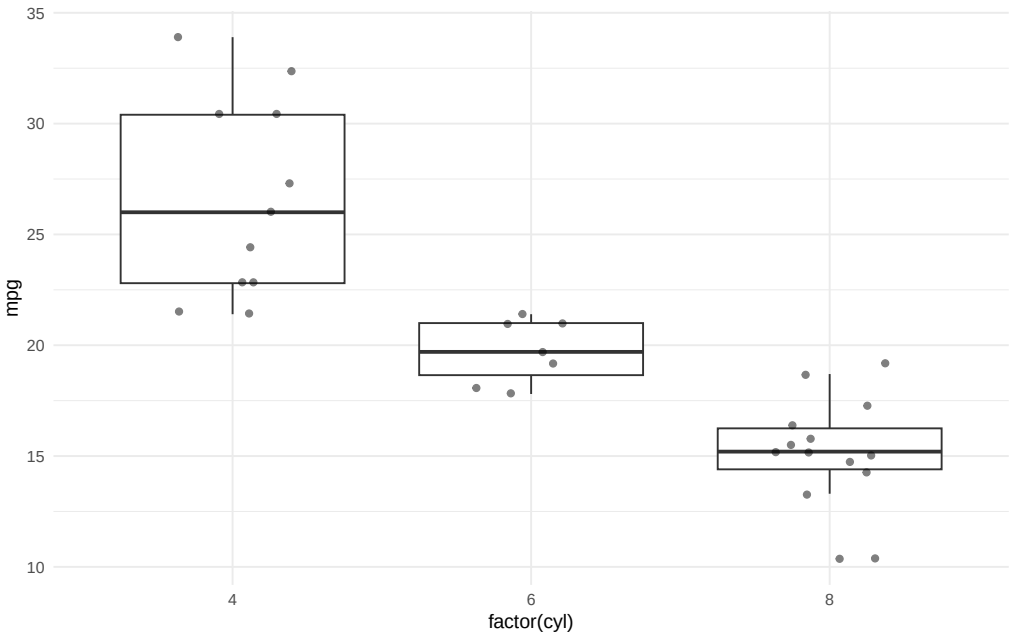
```
# Básico
ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot()
```



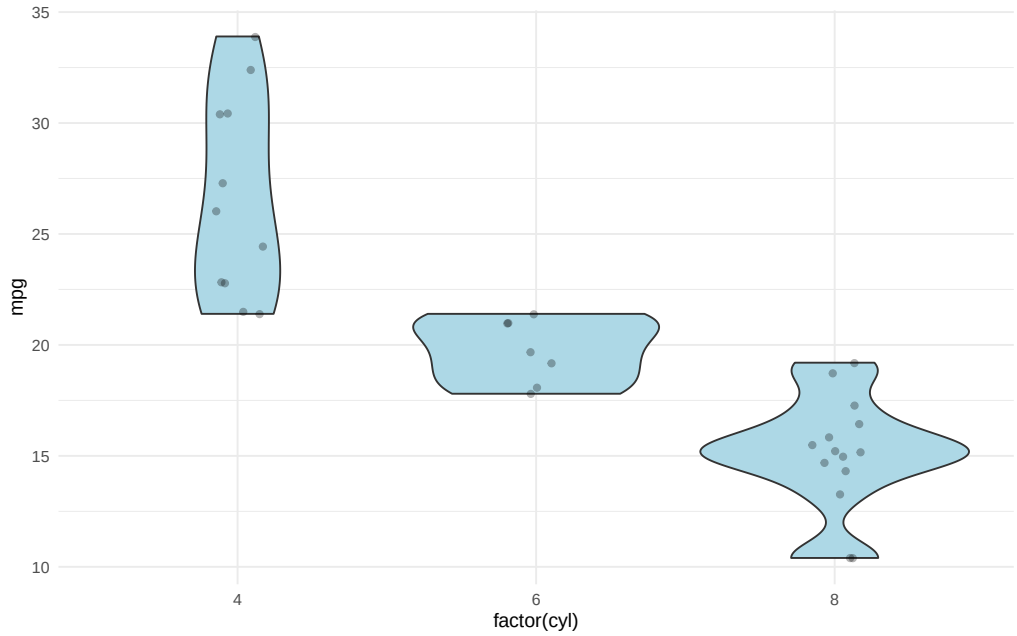

```
# Com cores
ggplot(mtcars, aes(x = factor(cyl), y = mpg, fill = factor(cyl))) +
  geom_boxplot() +
  labs(
    title = "Consumo por Número de Cilindros",
    x = "Cilindros",
    y = "Milhas por Galão",
    fill = "Cilindros"
  ) +
  theme_minimal()
```




```
# Com pontos (jitter)
ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot(outlier.shape = NA) +  # Remove outliers do boxplot
  geom_jitter(alpha = 0.5, width = 0.2) +  # Adiciona pontos
  theme_minimal()
```



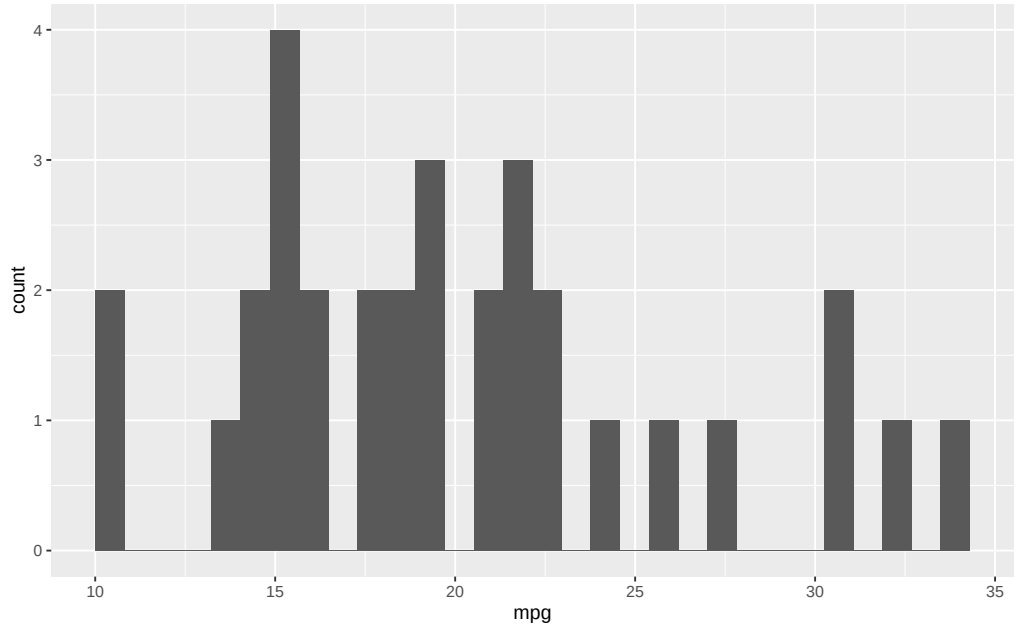

```
# Violin plot (alternativa)
ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_violin(fill = "lightblue") +
  geom_jitter(alpha = 0.3, width = 0.1) +
  theme_minimal()
```



45.5 2.5 Histograma e Densidade

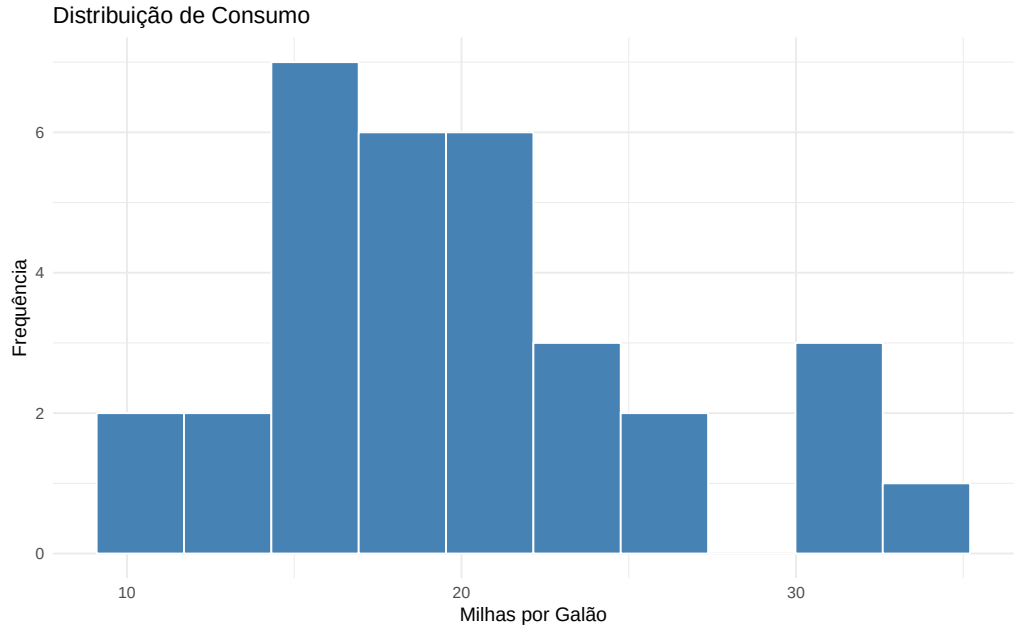
Histograma mostra distribuição de variável contínua.

```
# Histograma básico
ggplot(mtcars, aes(x = mpg)) +
  geom_histogram()
```

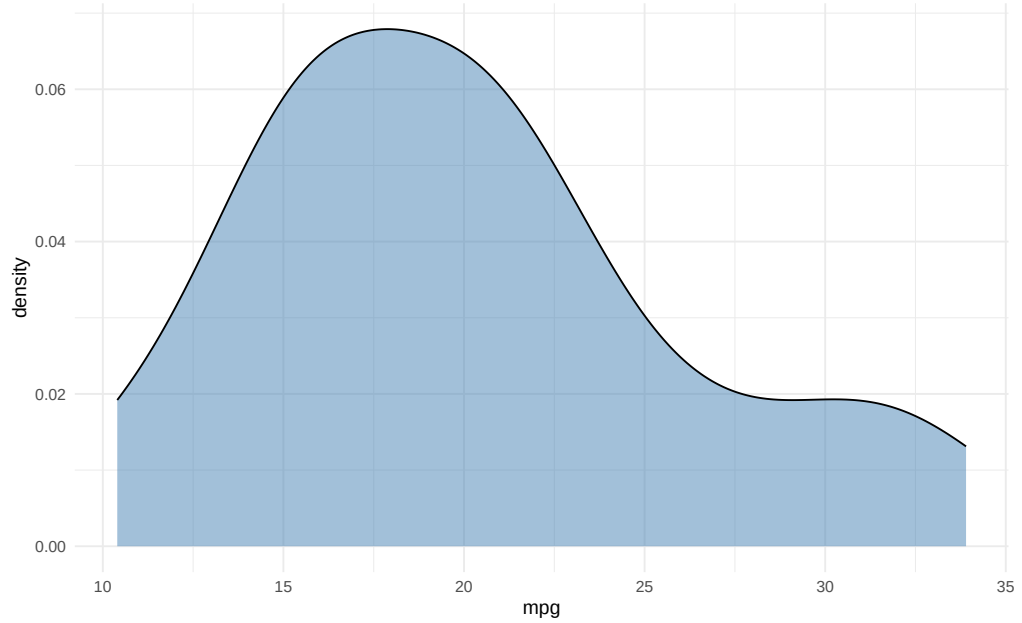



```
# Customizar bins
ggplot(mtcars, aes(x = mpg)) +
  geom_histogram(bins = 10, fill = "steelblue", color = "white") +
```

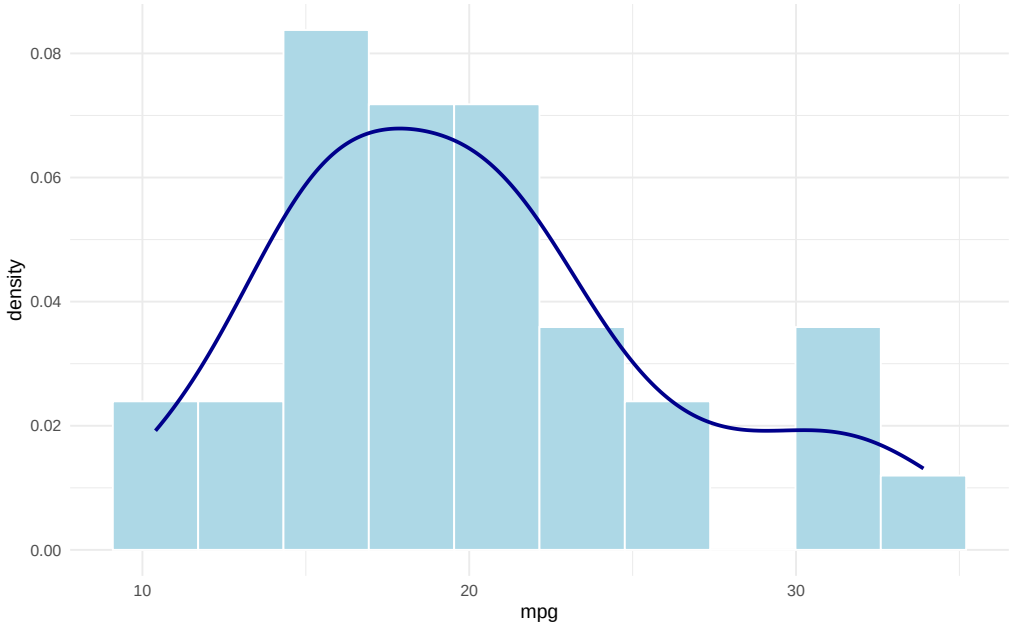
```
  labs(
    title = "Distribuição de Consumo",
    x = "Milhas por Galão",
    y = "Frequência"
  ) +
  theme_minimal()
```



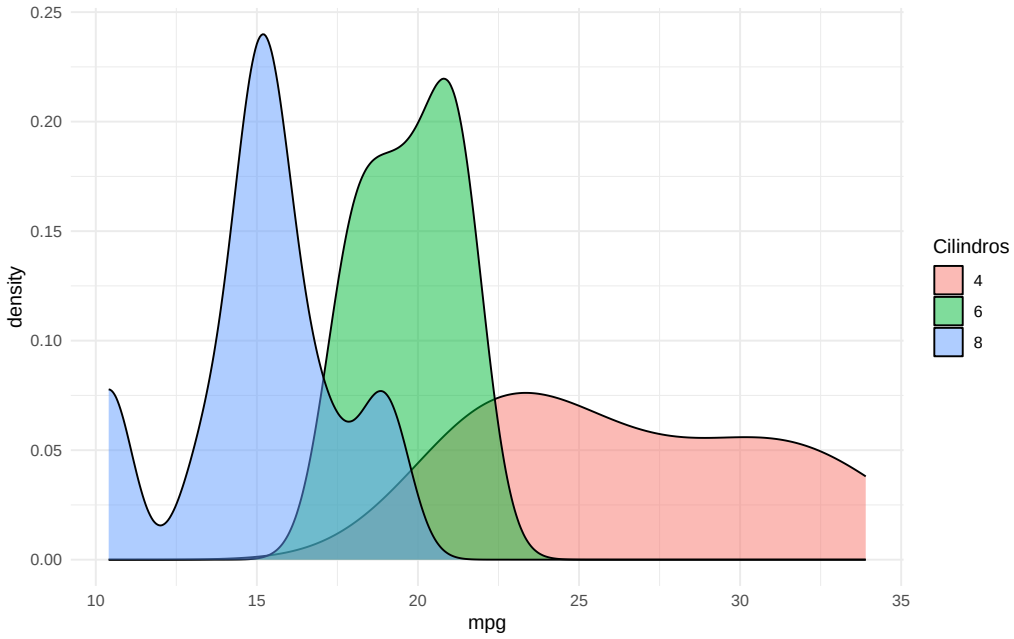

```
# Densidade
ggplot(mtcars, aes(x = mpg)) +
  geom_density(fill = "steelblue", alpha = 0.5) +
  theme_minimal()
```



```r
# Histograma + densidade
ggplot(mtcars, aes(x = mpg)) +
  geom_histogram(aes(y = after_stat(density)), bins = 10,
                 fill = "lightblue", color = "white") +
  geom_density(color = "darkblue", linewidth = 1) +
  theme_minimal()
```



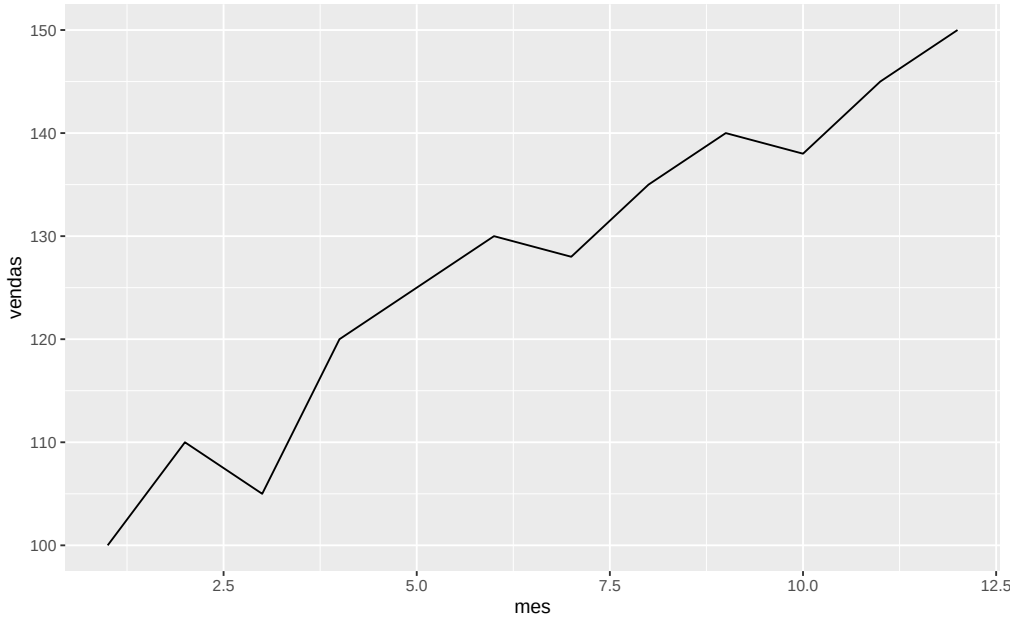

```r
# Por grupo
ggplot(mtcars, aes(x = mpg, fill = factor(cyl))) +
  geom_density(alpha = 0.5) +
  labs(fill = "Cilindros") +
  theme_minimal()
```



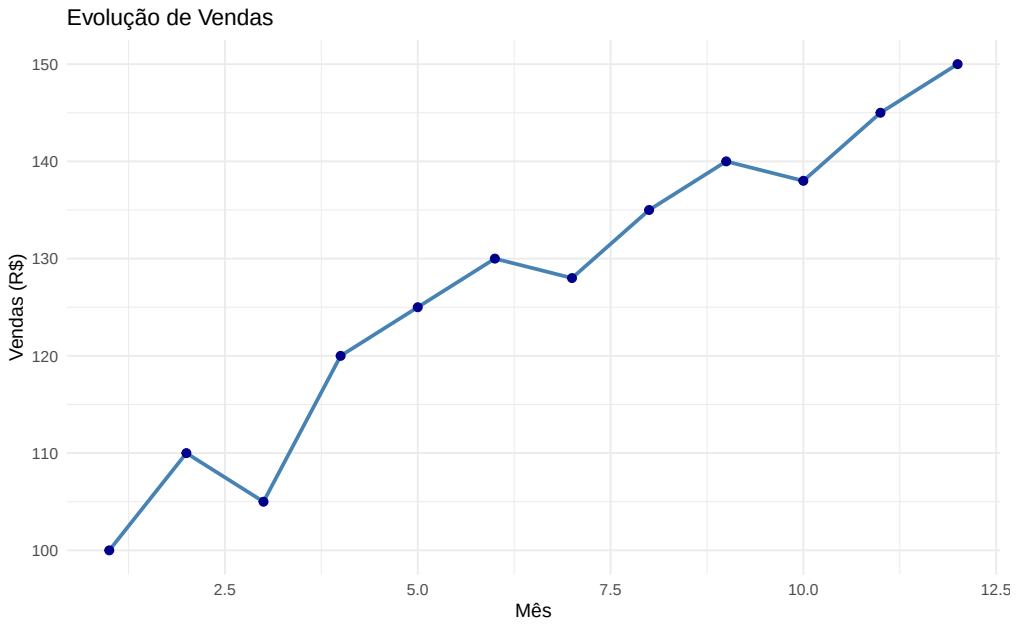
45.6 2.6 Gráfico de Linhas

Linhas mostram tendências ao longo do tempo.

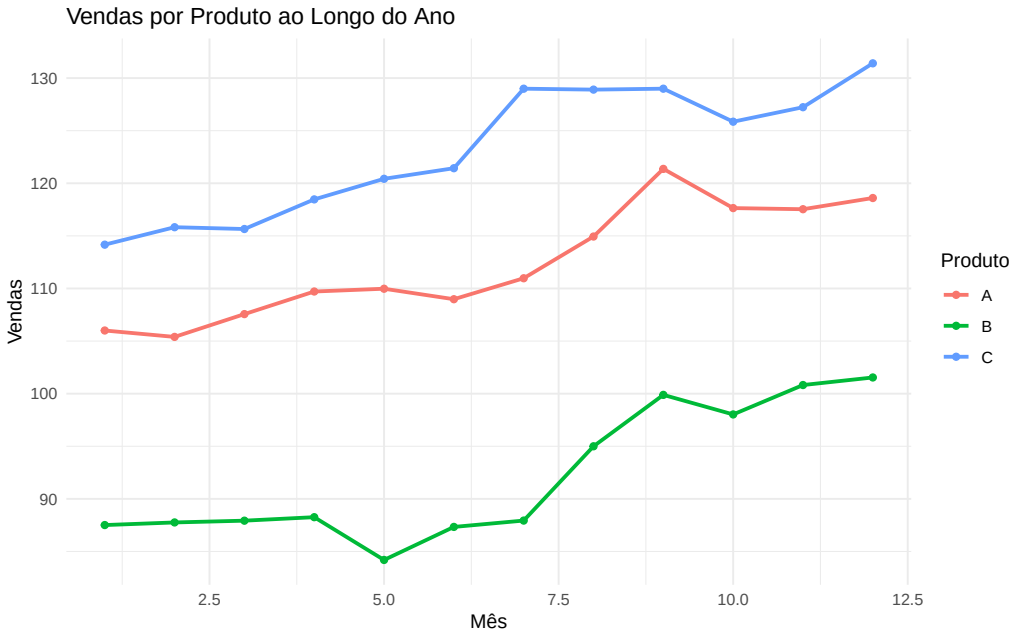
```
# Dados temporais
vendas_tempo <- tibble(
  mes = 1:12,
  vendas = c(100, 110, 105, 120, 125, 130, 128, 135, 140, 138, 145, 150)
)

# Básico
ggplot(vendas_tempo, aes(x = mes, y = vendas)) +
  geom_line()
```



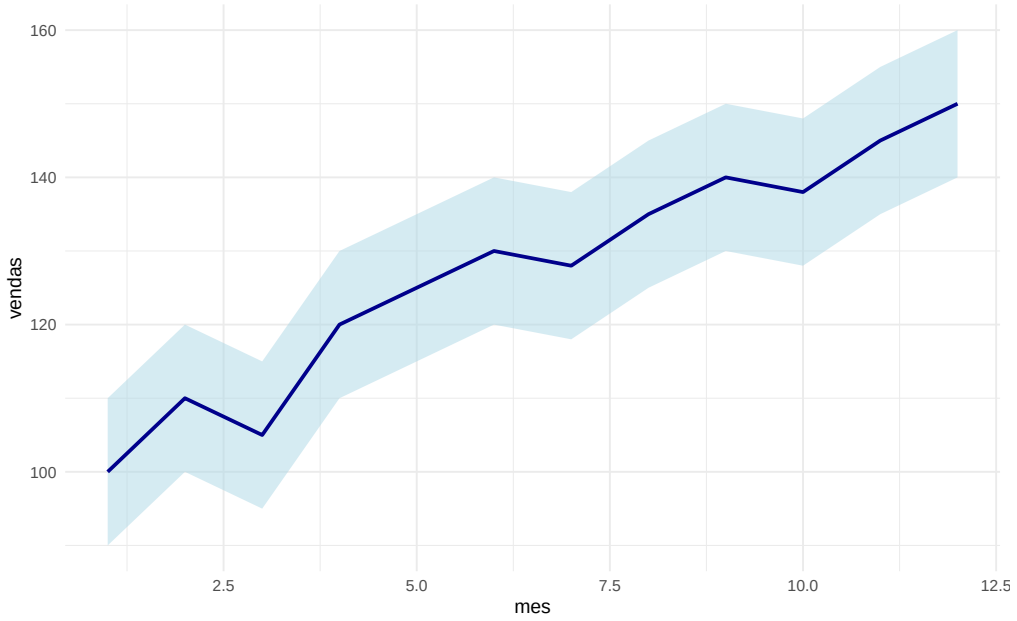

```
# Com pontos
ggplot(vendas_tempo, aes(x = mes, y = vendas)) +
  geom_line(color = "steelblue", linewidth = 1) +
  geom_point(color = "darkblue", size = 2) +
  labs(
    title = "Evolução de Vendas",
    x = "Mês",
    y = "Vendas (R$)"
  ) +
  theme_minimal()
```




```r
# Múltiplas linhas
vendas_produtos <- tibble(
  mes = rep(1:12, 3),
  produto = rep(c("A", "B", "C"), each = 12),
  vendas = c(
    100 + cumsum(rnorm(12, 2, 3)),
    90 + cumsum(rnorm(12, 2.5, 3)),
    110 + cumsum(rnorm(12, 1.5, 3))
  )
)

ggplot(vendas_produtos, aes(x = mes, y = vendas, color = produto)) +
  geom_line(linewidth = 1) +
  geom_point() +
  labs(
    title = "Vendas por Produto ao Longo do Ano",
    x = "Mês",
    y = "Vendas",
    color = "Produto"
  ) +
  theme_minimal()
```



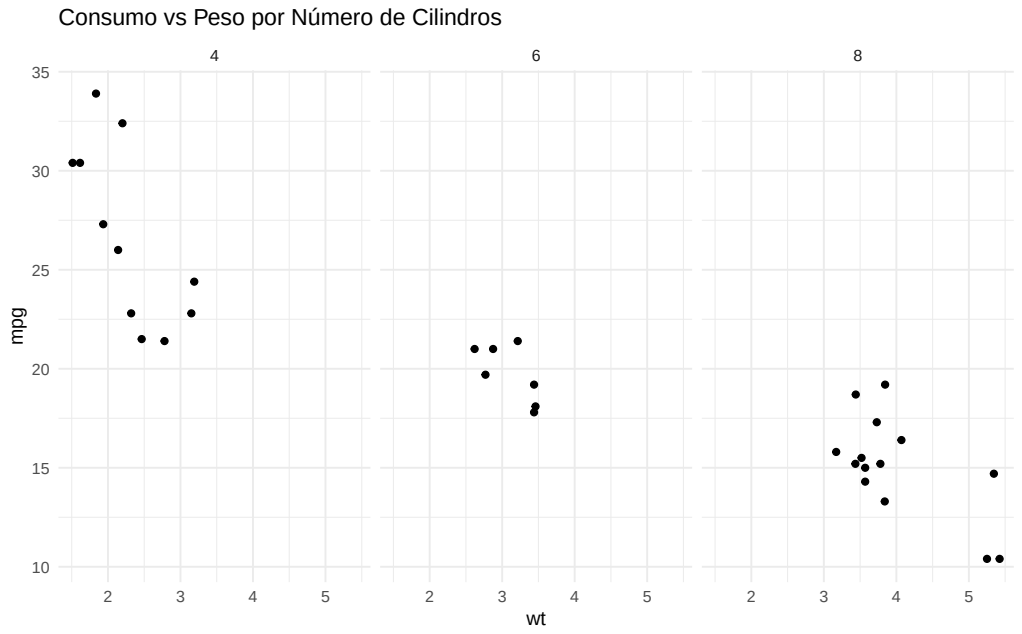

```
# Com área (ribbon)
vendas_ic <- vendas_tempo %>%
  mutate(
    ic_lower = vendas - 10,
    ic_upper = vendas + 10
  )

ggplot(vendas_ic, aes(x = mes, y = vendas)) +
  geom_ribbon(aes(ymin = ic_lower, ymax = ic_upper),
              fill = "lightblue", alpha = 0.5) +
  geom_line(color = "darkblue", linewidth = 1) +
  theme_minimal()
```



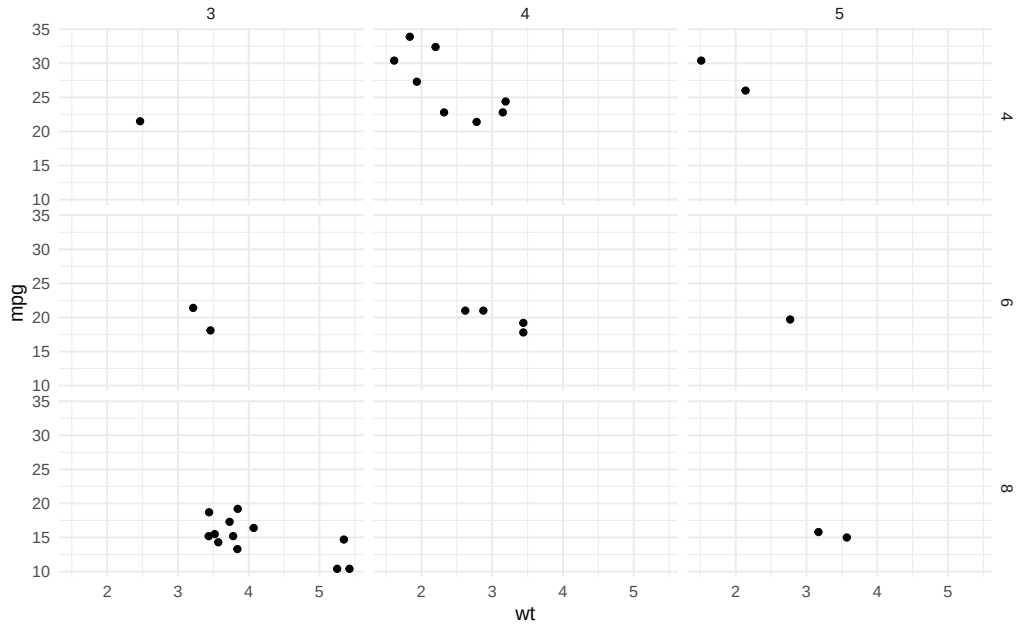
45.7 2.7 Facetas

Facetas criam múltiplos gráficos para diferentes subgrupos.

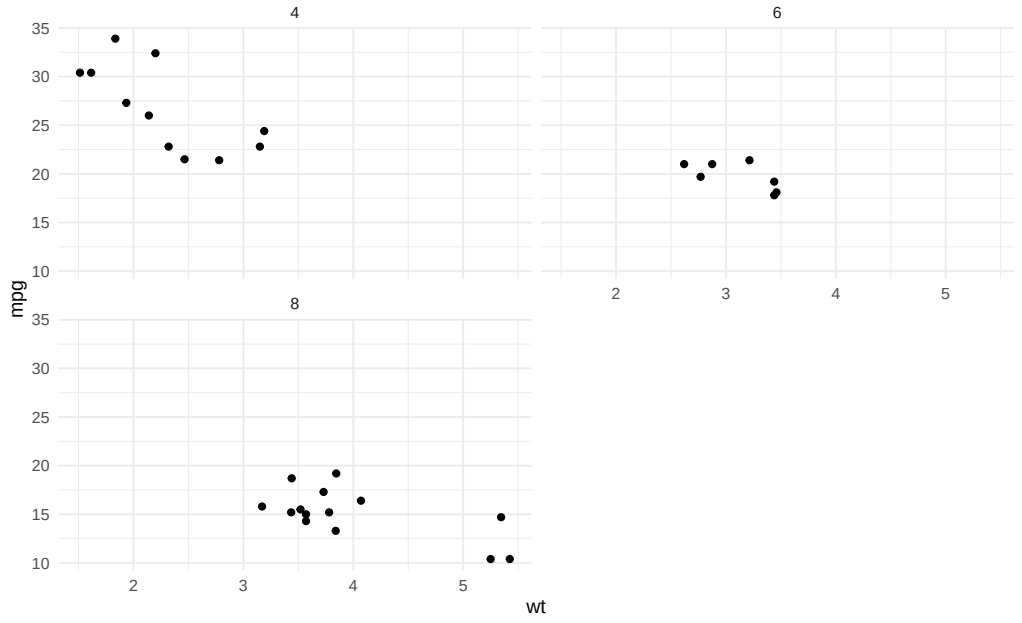
```
# facet_wrap() - uma variável
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  facet_wrap(~cyl) +
  labs(title = "Consumo vs Peso por Número de Cilindros") +
  theme_minimal()
```



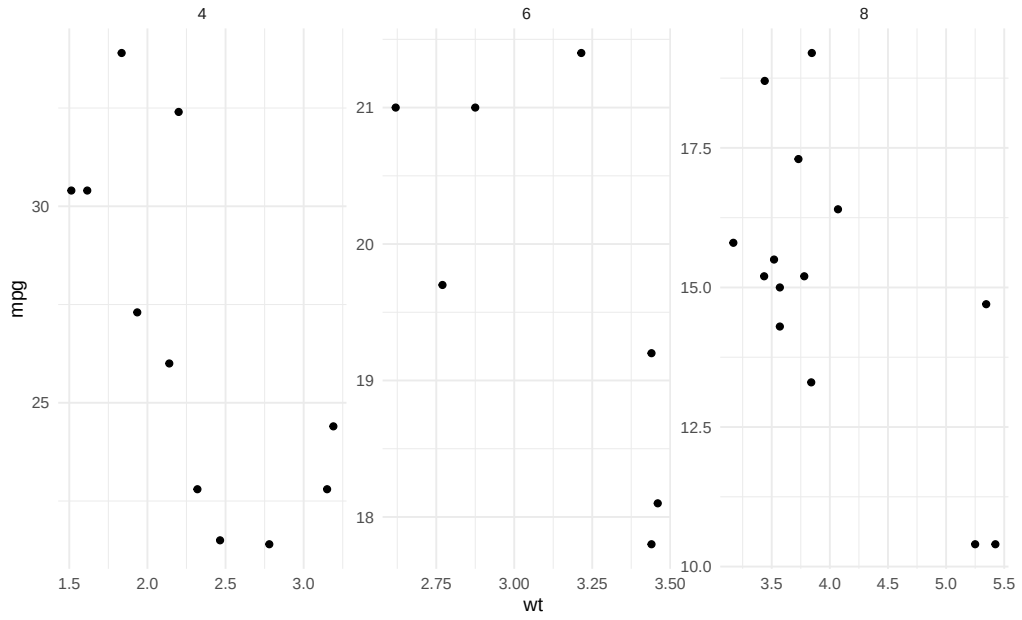

```
# facet_grid() - duas variáveis
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  facet_grid(cyl ~ gear) +
  theme_minimal()
```




```r
# Customizar layout
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  facet_wrap(~cyl, ncol = 2) +  # 2 colunas
  theme_minimal()
```



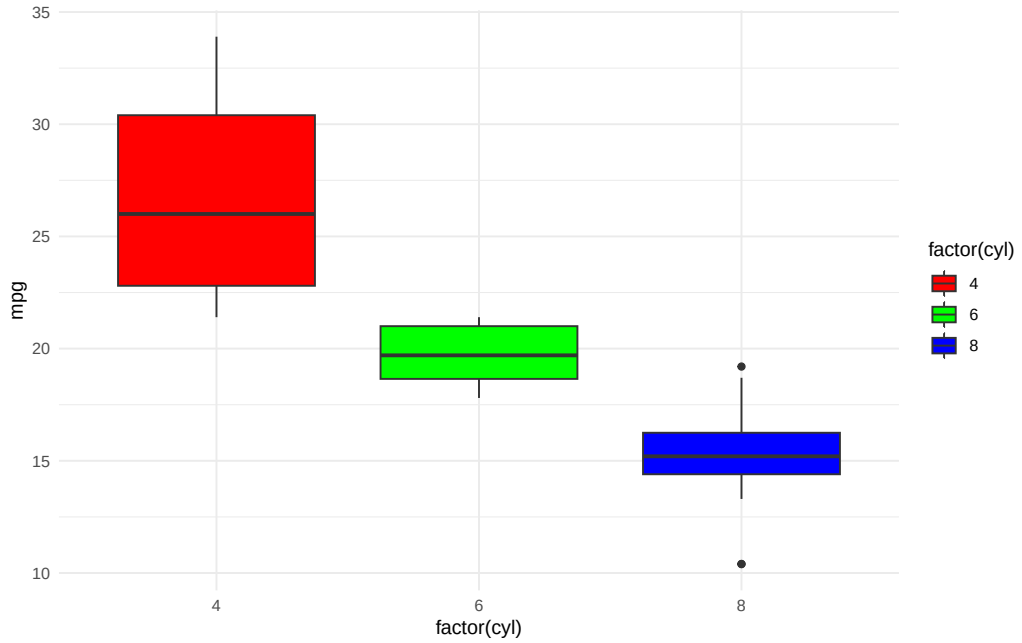

```r
# Escalas livres
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  facet_wrap(~cyl, scales = "free") +  # Escalas independentes
  theme_minimal()
```



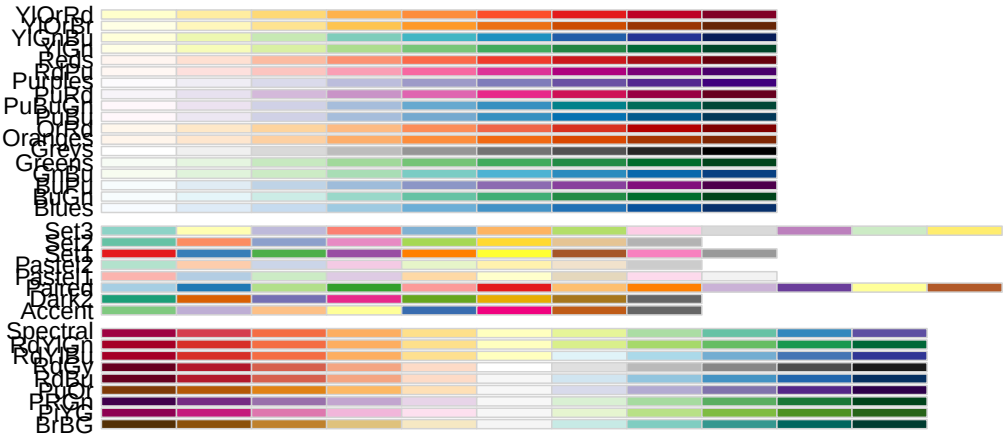
45.8 2.8 Personalização

45.8.1 Cores

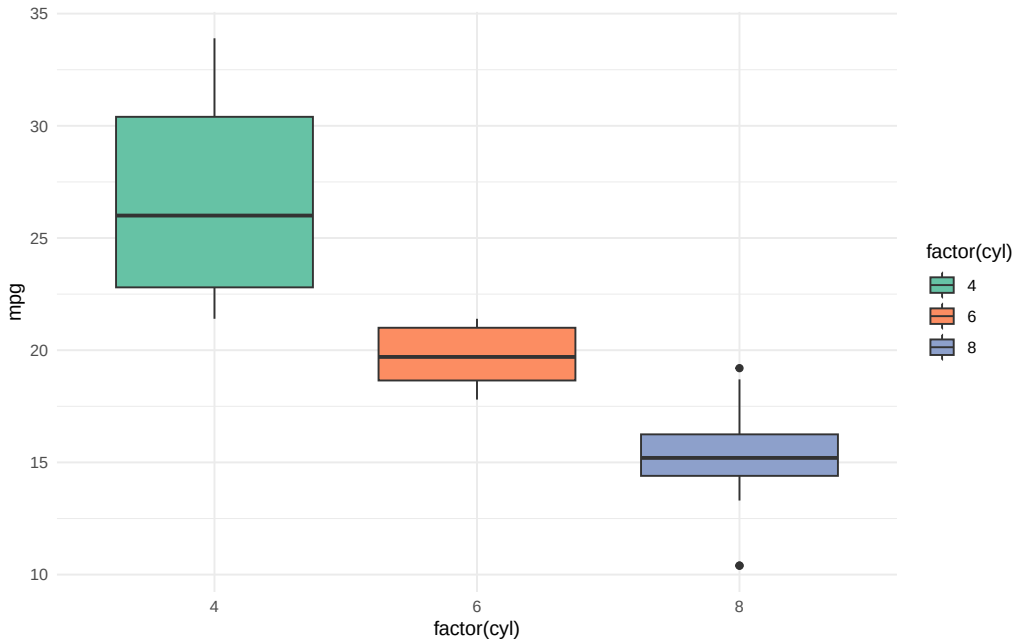
```
# Cores manuais
ggplot(mtcars, aes(x = factor(cyl), y = mpg, fill = factor(cyl))) +
  geom_boxplot() +
  scale_fill_manual(values = c("red", "green", "blue")) +
  theme_minimal()
```



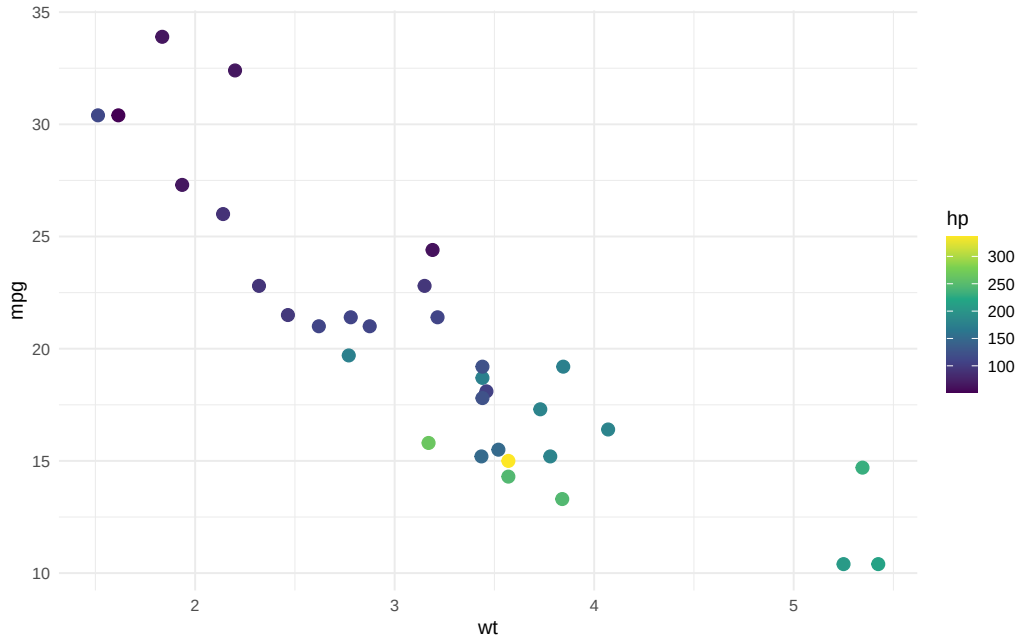
```r
# Paletas do RColorBrewer
library(RColorBrewer)
display.brewer.all()
```



```r
ggplot(mtcars, aes(x = factor(cyl), y = mpg, fill = factor(cyl))) +
  geom_boxplot() +
  scale_fill_brewer(palette = "Set2") +
  theme_minimal()
```

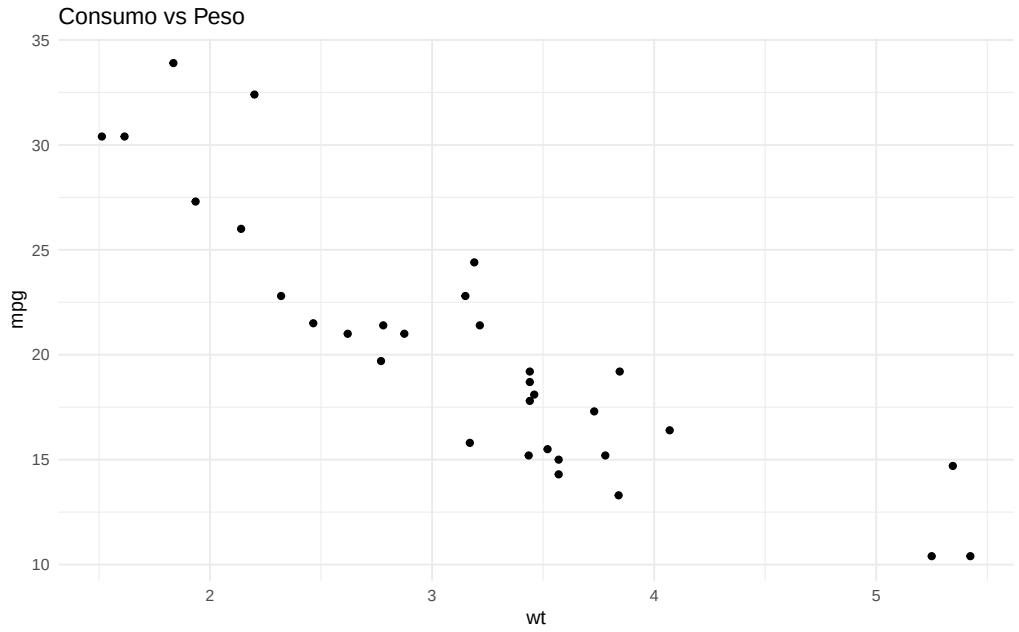



```r
# Paletas viridis (color-blind friendly)
ggplot(mtcars, aes(x = wt, y = mpg, color = hp)) +
  geom_point(size = 3) +
  scale_color_viridis_c() +  # _c = contínua, _d = discreta
  theme_minimal()
```

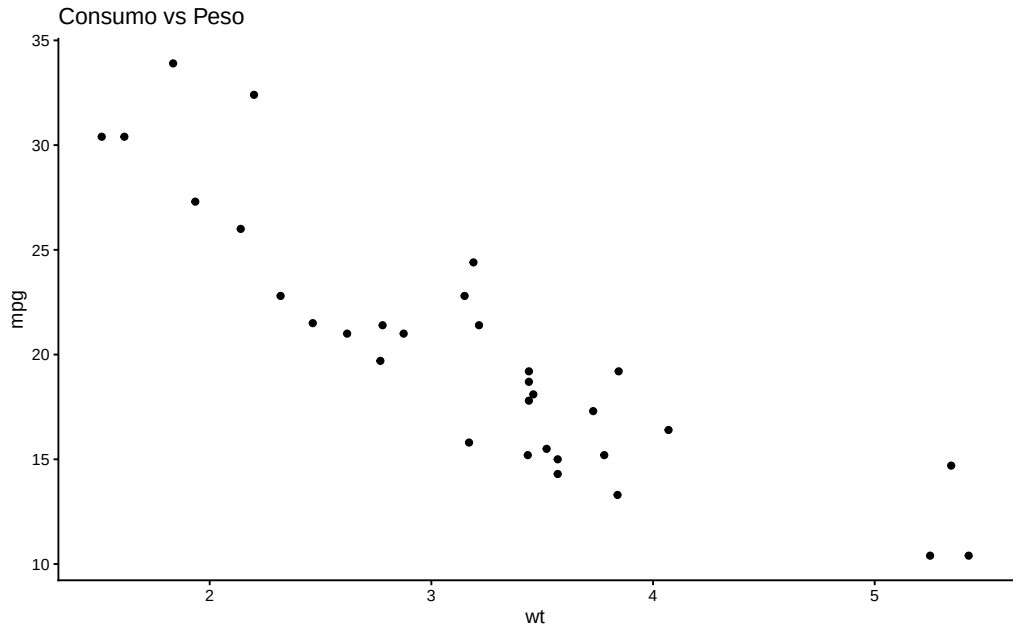


45.8.2 Temas

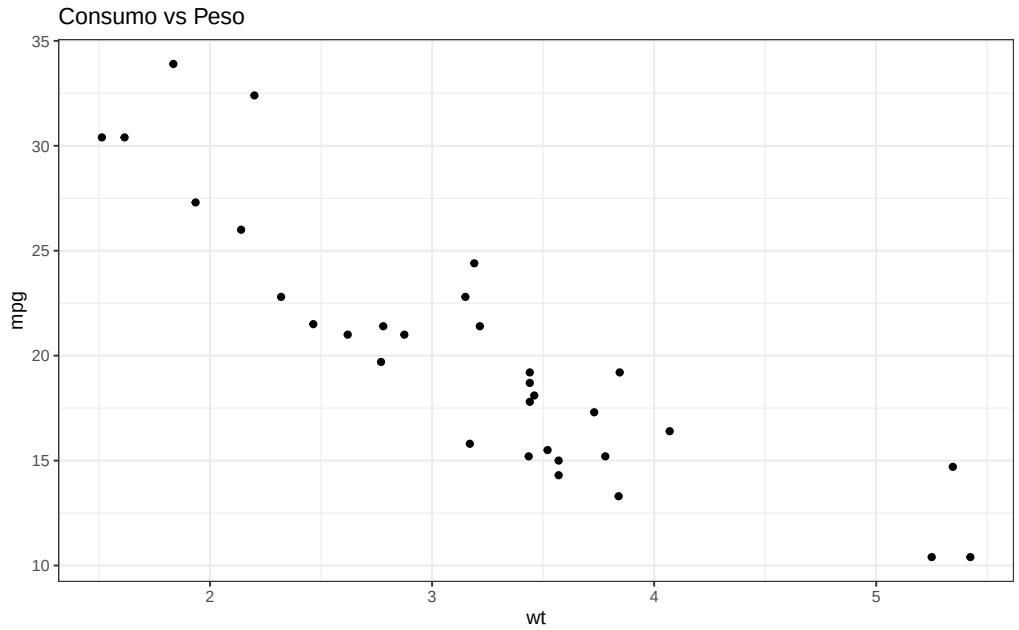
```r
p <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  labs(title = "Consumo vs Peso")

# Temas disponíveis
p + theme_minimal()
```



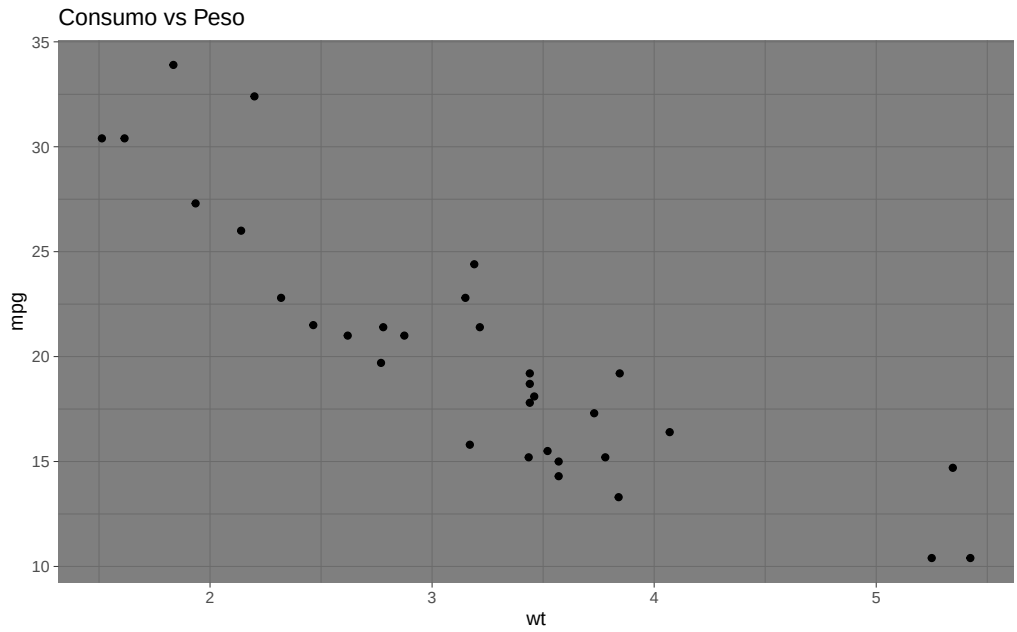
```
p + theme_classic()
```



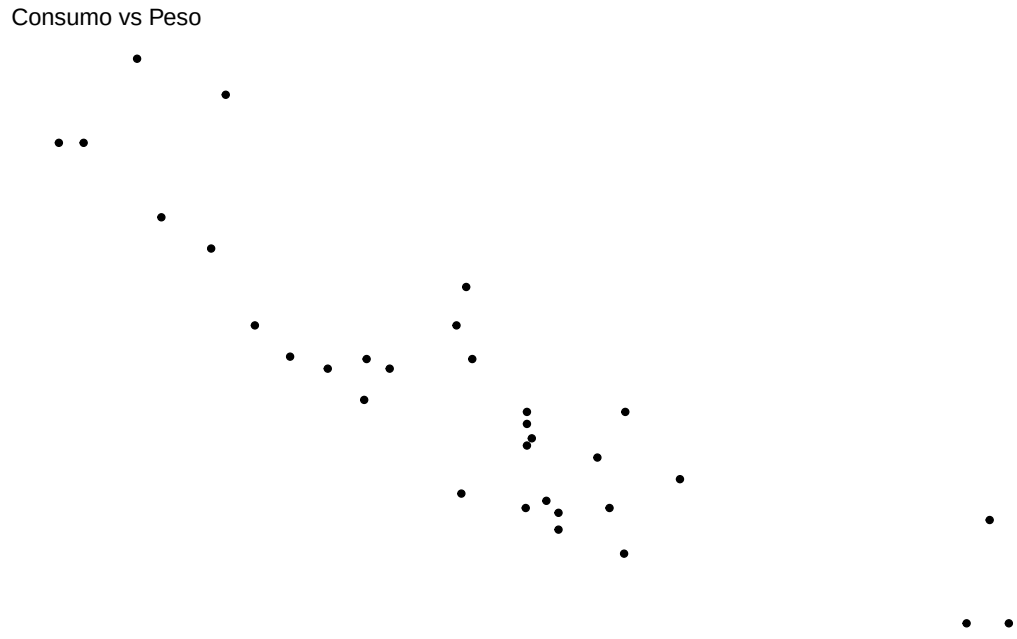

```
p + theme_bw()
```



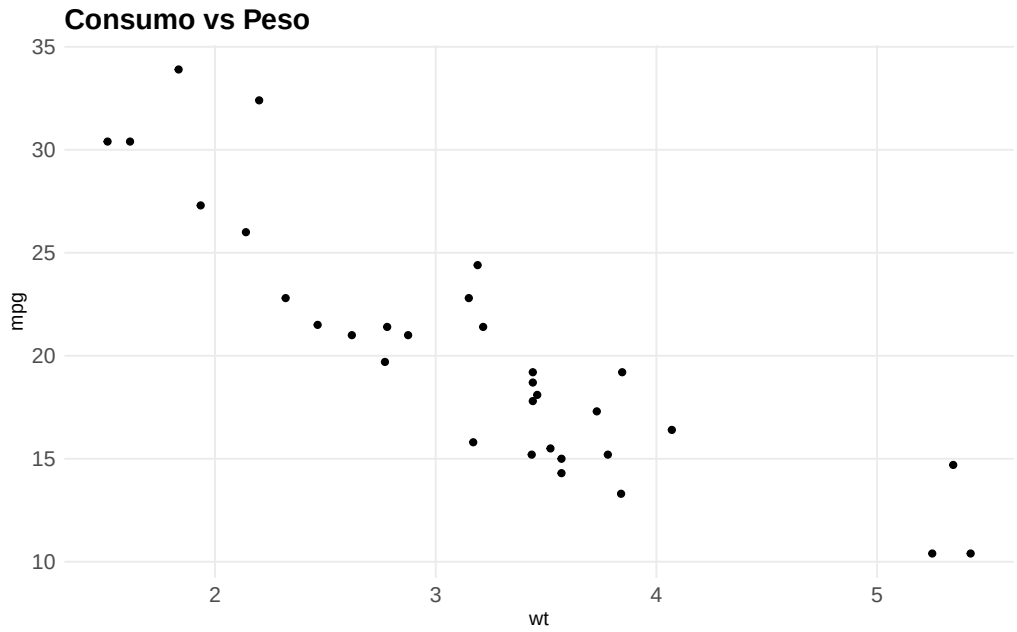

```
p + theme_dark()
```




```
p + theme_void()
```

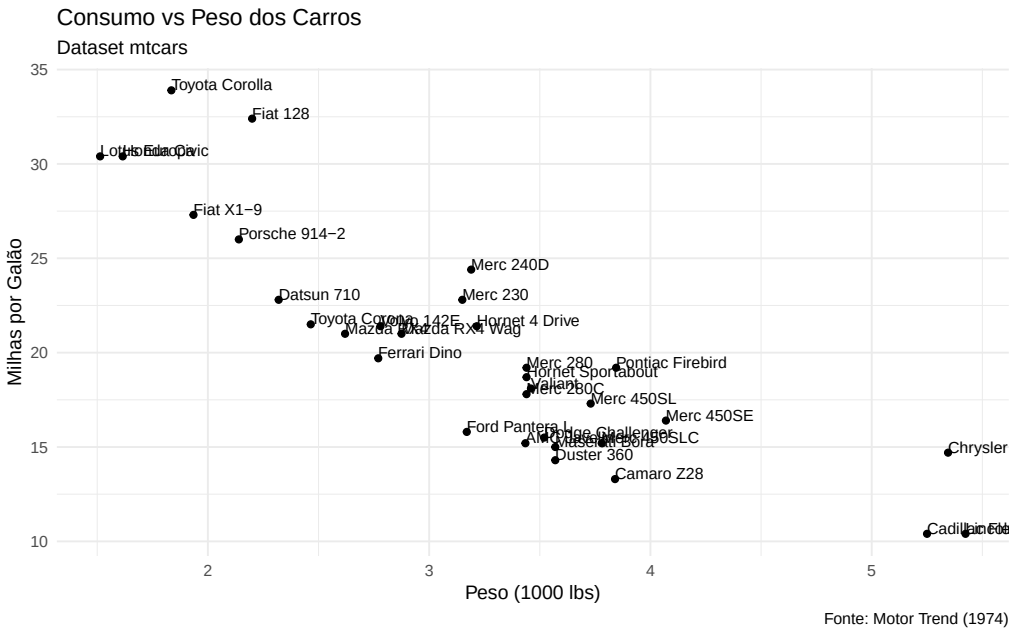



```
# Customizar tema
p + theme_minimal() +
  theme(
    plot.title = element_text(size = 16, face = "bold"),
    axis.text = element_text(size = 12),
    panel.grid.minor = element_blank()
  )
```

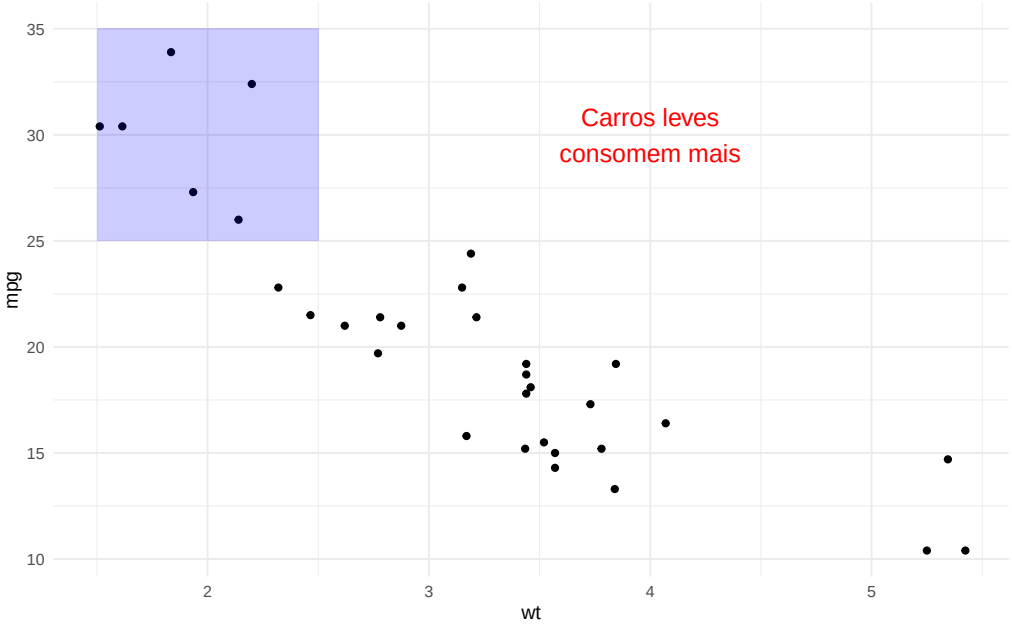


45.8.3 Labels e Anotações

```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  geom_text(aes(label = rownames(mtcars)),
            hjust = 0, vjust = 0, size = 3) +
  labs(
    title = "Consumo vs Peso dos Carros",
    subtitle = "Dataset mtcars",
    x = "Peso (1000 lbs)",
    y = "Milhas por Galão",
    caption = "Fonte: Motor Trend (1974)"
  ) +
  theme_minimal()
```




```r
# Anotação manual
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  annotate("text", x = 4, y = 30, label = "Carros leves\nconsomem mais",
           color = "red", size = 5) +
  annotate("rect", xmin = 1.5, xmax = 2.5, ymin = 25, ymax = 35,
           alpha = 0.2, fill = "blue") +
  theme_minimal()
```



45.9 2.9 Salvando Gráficos

```r
# Criar gráfico
p <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Consumo vs Peso") +
  theme_minimal()

# Salvar com ggsave()
ggsave(
  filename = here("output", "figures", "grafico_consumo.png"),
  plot = p,
  width = 8,       # Largura em polegadas
  height = 6,      # Altura em polegadas
  dpi = 300        # Resolução (300 dpi para publicação)
)

# Diferentes formatos
ggsave(here("output", "grafico.png"), p)   # PNG
ggsave(here("output", "grafico.pdf"), p)   # PDF (vetorial)
ggsave(here("output", "grafico.svg"), p)   # SVG (vetorial)
ggsave(here("output", "grafico.jpg"), p)   # JPEG

# Ajustar tamanho
ggsave(here("output", "grafico_wide.png"), p,
       width = 12, height = 4)  # Formato wide

# Salvar último gráfico
ggplot(mtcars, aes(x = wt, y = mpg)) + geom_point()
ggsave(here("output", "ultimo_grafico.png"))  # Salva o último
```

---

46 Exercícios Práticos

46.1 Exercício 1: I/O de Dados

```r
# a) Crie um tibble com dados fictícios (nome, idade, cidade, salário)
#    de 10 pessoas


# b) Salve como CSV em data/processed/


# c) Salve como Excel com duas sheets: "Dados" e "Resumo"
```

```r
# d) Salve como RDS


# e) Leia cada arquivo de volta e compare
```

46.2 Exercício 2: Visualizações Básicas

```r
# Use o dataset 'iris' (nativo do R)
glimpse(iris)

# a) Crie gráfico de dispersão: Sepal.Length vs Sepal.Width
#    colorido por Species


# b) Crie boxplot de Petal.Length por Species


# c) Crie histograma de Sepal.Length


# d) Salve os 3 gráficos em PNG (alta resolução)
```

46.3 Exercício 3: Gráficos Avançados

```r
# Use o dataset 'diamonds' (ggplot2)
glimpse(diamonds)

# a) Crie gráfico de densidade do preço por qualidade do corte (cut)


# b) Crie gráfico de barras: média de preço por cor (color)


# c) Crie facetas: dispersão price vs carat, uma faceta por clarity


# d) Customize com tema, cores e títulos profissionais
```

46.4 Exercício 4: Análise Completa

```r
# Dataset de vendas (criar fictício)
set.seed(123)
vendas <- tibble(
  data = seq(as.Date("2024-01-01"), by = "day", length.out = 365),
  produto = sample(c("A", "B", "C"), 365, replace = TRUE),
  vendas = round(rnorm(365, 1000, 200)),
```

```
  regiao = sample(c("Norte", "Sul", "Leste", "Oeste"), 365, replace = TRUE)
)

# a) Salve o dataset em CSV


# b) Crie gráfico de linha: vendas ao longo do tempo por produto


# c) Crie boxplot: vendas por região


# d) Crie facetas: densidade de vendas por região


# e) Salve os gráficos em uma pasta 'output/figures/'
```

46.5 Exercício 5: Projeto Integrado

```
# Projeto completo: análise de desempenho de alunos

# 1. Crie dataset de 50 alunos com:
#    - nome, curso, nota_p1, nota_p2, frequência, bolsista


# 2. Calcule média final e status (aprovado/reprovado)


# 3. Salve dados limpos em processed/


# 4. Crie 4 visualizações:
#    - Distribuição de notas
#    - Média por curso
#    - Relação entre frequência e nota
#    - Comparação bolsista vs não-bolsista


# 5. Salve todas as visualizações profissionalmente


# 6. Use IA (ChatGPT ou Claude) para sugerir melhorias!
```

---

47 Galeria de Gráficos

47.1 Quando usar cada tipo?

```
# DISPERSÃO: relação entre 2 variáveis contínuas
# Exemplo: altura vs peso, preço vs tamanho

# LINHAS: evolução temporal, tendências
# Exemplo: vendas mensais, temperatura diária

# BARRAS: comparar categorias
# Exemplo: vendas por produto, população por país

# BOXPLOT: distribuição, outliers, comparar grupos
# Exemplo: salário por profissão, notas por turma

# HISTOGRAMA: distribuição de uma variável
# Exemplo: distribuição de idades, frequência de valores

# DENSIDADE: distribuição suave, comparar grupos
# Exemplo: distribuição de renda por região

# FACETAS: comparar subgrupos
# Exemplo: mesma análise para diferentes categorias
```

---

48 Commit no GitHub

Versione seu progresso:

```
git add .
git commit -m "Dia 4: adiciona I/O e visualização com ggplot2"
git push
```

---

49 Para Casa

1. **Organizar** seu projeto com estrutura profissional
2. **Praticar** leitura de seus próprios dados
3. **Criar** pelo menos 5 visualizações diferentes
4. **Ler** R for Data Science - Data Visualization
5. **Explorar** R Graph Gallery para inspiração
6. **Usar** IA para melhorar suas visualizações

---

50 Recursos Adicionais

50.1 Documentação

- ggplot2 Cheat Sheet
- ggplot2 Book
- R Graph Gallery
- Data-to-Viz - Escolher gráfico certo

50.2 Extensões ggplot2

- **ggthemes**: Temas adicionais
- **patchwork**: Combinar múltiplos gráficos
- **ggrepel**: Labels sem sobreposição
- **gganimate**: Gráficos animados
- **plotly**: Gráficos interativos

```r
# install.packages(c("ggthemes", "patchwork", "ggrepel"))
library(ggthemes)
library(patchwork)

# Tema Economist
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_economist()

# Combinar gráficos
p1 <- ggplot(mtcars, aes(wt, mpg)) + geom_point()
p2 <- ggplot(mtcars, aes(hp, mpg)) + geom_point()
p1 + p2  # Lado a lado
```

---

51 Dúvidas Frequentes

P: CSV ou Excel?
R: CSV para dados simples, Excel quando precisa de múltiplas sheets ou formatação.

P: Por que usar here()?
R: Portabilidade! Seu código funciona em qualquer computador sem mudanças.

P: Qual resolução usar para gráficos?
R: 300 dpi para publicação, 150 dpi para web, 72 dpi para apresentações.

P: Como escolher cores?
R: Use paletas viridis (color-blind friendly) ou ColorBrewer. Evite vermelho-verde juntos.

P: Gráfico muito carregado, o que fazer?
R: Simplifique! Menos é mais. Use facetas para separar informação.

---

52 Conclusão do Dia 4

Parabéns! Você completou o Dia 4 e agora domina:

- Leitura e escrita de dados (CSV, Excel, RDS)
- Organização profissional de projetos
- Caminhos relativos com here()
- Gramática de gráficos do ggplot2
- Criação de visualizações profissionais
- Personalização de gráficos
- Salvamento em alta qualidade
- Escolha do gráfico adequado

Amanhã: RMarkdown, relatórios reprodutíveis e projeto final!

Última atualização: 2025-10-07
Contato: junqueiravinicius@hotmail.com
Repositório: https://github.com/viniciusjunqueira/curso-r-github-ia
Lattes: http://lattes.cnpq.br/4686677580216927

53 Parte 5: Relat rios e Projeto Final

---

54 Objetivos do Dia 5

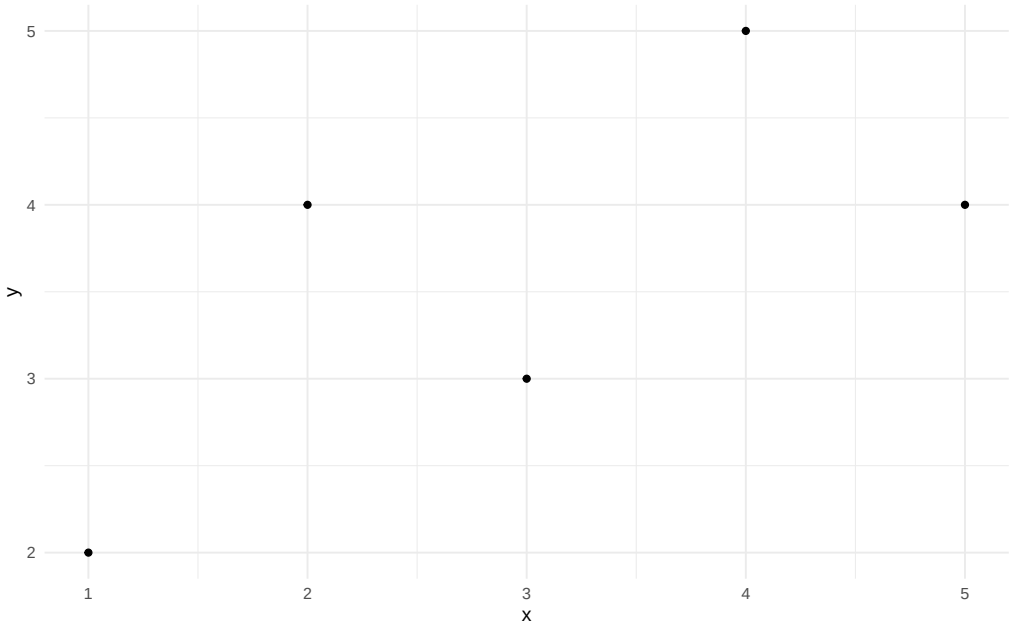
Ao final desta aula, você será capaz de:

- Criar documentos reprodutíveis com RMarkdown
- Entender a estrutura YAML, Markdown e chunks
- Gerar relatórios em HTML, PDF e Word
- Usar código inline para resultados dinâmicos
- Organizar projetos finais profissionalmente
- Criar README.md completo no GitHub
- Configurar .gitignore adequadamente
- Entender conceitos básicos de branches
- Usar IA para documentação e revisão
- Continuar aprendendo R de forma autônoma

---

55 Revisão do Curso

```r
library(tidyverse)
library(here)

# Dia 1: Fundamentos
vetor <- c(1, 2, 3, 4, 5)
media <- mean(vetor)

# Dia 2: Funções
calcular_imc <- function(peso, altura) {
  peso / altura^2
}

# Dia 3: Manipulação
dados <- tibble(x = 1:5, y = c(2, 4, 3, 5, 4))
resumo <- dados %>%
  summarize(media_y = mean(y))

# Dia 4: Visualização
ggplot(dados, aes(x, y)) +
  geom_point() +
  theme_minimal()
```



56 Parte 1: RMarkdown e Documentos Reprodutíveis (19h00 - 20h30)

56.1 1.1 O que é Programação Literária?

Programação literária combina código, resultados e narrativa em um único documento.

56.1.1 Benefícios

- **Reprodutibilidade:** Qualquer pessoa pode recriar sua análise
- **Transparência:** Código e resultados visíveis
- **Eficiência:** Atualiza automaticamente quando dados mudam
- **Comunicação:** Explica o processo de análise
- **Documentação:** Registro completo do que foi feito

56.1.2 RMarkdown vs Quarto

```
# RMarkdown (tradicional)
# - Foco em R
# - Maduro e estável
# - Grande comunidade

# Quarto (novo)
# - Multi-linguagem (R, Python, Julia)
# - Mais recursos modernos
# - Sucessor do RMarkdown
```

```
# Neste curso: RMarkdown (mais comum ainda)
# Princípios são os mesmos!
```

56.2 1.2 Criando um Documento RMarkdown

56.2.1 No RStudio

1. **File → New File → R Markdown**
2. Escolha título, autor, formato
3. Clique OK

56.2.2 Estrutura Básica

````
---
title: "Meu Relatório"
author: "Seu Nome"
date: "2025-10-07"
output: html_document
---

# Introdução

Este é um documento RMarkdown.


```r
# Código R aqui
x <- 1:10
mean(x)
```

```
#> [1] 5.5
```

## Resultados

A média é 5.5.
````

56.3 1.3 Componentes de um Documento

56.3.1 YAML Header

O **YAML** (Yet Another Markup Language) configura o documento:

```yaml
---
title: "Análise de Vendas"
author: "Vinícius Junqueira"
date: "2025-10-07"
output:
  html_document:
    toc: true
    toc_float: true
    theme: flatly
    code_folding: hide
  pdf_document:
    toc: true
    latex_engine: xelatex
---
```

56.3.2 Opções Comuns

```yaml
# HTML
output:
  html_document:
    toc: true              # Índice
    toc_float: true        # Índice flutuante
    toc_depth: 3           # Profundidade
    number_sections: true  # Numerar seções
    theme: flatly          # Tema visual
    highlight: tango       # Syntax highlighting
    code_folding: hide     # Esconder código
    df_print: paged        # Tabelas paginadas

# PDF
output:
  pdf_document:
    toc: true
    number_sections: true
    latex_engine: xelatex  # Para Unicode
    keep_tex: false        # Manter arquivo .tex
```

56.4 1.4 Markdown Básico

Markdown é linguagem simples para formatação de texto.

56.4.1 Formatação de Texto

```
# Título Nível 1
## Título Nível 2
### Título Nível 3
```

```
**Negrito** ou __negrito__
*Itálico* ou _itálico_
~~Riscado~~

Texto normal com parágrafo.
Nova linha precisa de linha em branco.

- Item 1
- Item 2
  - Subitem 2.1
  - Subitem 2.2

1. Primeiro
2. Segundo
3. Terceiro
```

56.4.2 Links e Imagens

```
[Texto do link](https://example.com)
[R for Data Science](https://r4ds.hadley.nz/)

![Texto alternativo](caminho/imagem.png)
![Logo R](https://www.r-project.org/logo/Rlogo.png)
```

56.4.3 Outros Elementos

```
> Citação em bloco
> Pode ter múltiplas linhas

---

Linha horizontal

`código inline`

Código em bloco:
```

sem highlighting

```
Tabela:
| Coluna 1 | Coluna 2 |
|----------|----------|
| A        | 1        |
| B        | 2        |
```

56.5 1.5 Chunks de Código

Chunks são blocos de código R executáveis.

56.5.1 Sintaxe Básica

````
```r
# Código R aqui
x <- 1:10
mean(x)
```

```
#> [1] 5.5
```
````

56.5.2 Opções de Chunk

````
```r
# echo: mostrar código
# eval: executar código
# warning: mostrar warnings
# message: mostrar messages
# error: continuar se houver erro
# include: incluir no documento

library(tidyverse)
data <- read_csv("dados.csv")
```
````

56.5.3 Opções Comuns

```
# Executar mas não mostrar código
# {r, echo=FALSE}

# Mostrar código mas não executar
# {r, eval=FALSE}

# Não incluir no documento (útil para setup)
# {r, include=FALSE}

# Tamanho de figuras
# {r, fig.width=10, fig.height=6}

# Legenda de figura
# {r, fig.cap="Gráfico de dispersão"}
```

```
# Cache (salvar resultado)
# {r, cache=TRUE}
```

56.5.4 Chunk de Setup

56.6 1.6 Código Inline

Código inline insere resultados no texto.

```
A média das idades é 27.6666667 anos.

O dataset tem 32 observações.

Hoje é 2025-10-07.

O valor de pi é aproximadamente 3.14.
```

Resultado:

A média das idades é 27.6666667 anos.

56.7 1.7 Tabelas

56.7.1 Tabelas Simples

```
# Tabela básica do R
head(mtcars, 3)
```

```
#>                mpg cyl disp  hp drat    wt  qsec vs am gear carb
#> Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
#> Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
#> Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

```
# knitr::kable() - mais bonito
library(knitr)
kable(head(mtcars, 3), caption = "Primeiras linhas de mtcars")
```

Table 11: Primeiras linhas de mtcars

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1

Motor Trend Car Road Tests
Primeiras 5 observações

mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
21.0	6	160.0	110.0	3.9	2.6	16.46	0	1	4	4
21.0	6	160.0	110.0	3.9	2.9	17.02	0	1	4	4
22.8	4	108.0	93.0	3.9	2.3	18.61	1	1	4	1
21.4	6	258.0	110.0	3.1	3.2	19.44	1	0	3	1
18.7	8	360.0	175.0	3.1	3.4	17.02	0	0	3	2

```r
# Com alinhamento e formato
kable(head(mtcars, 3),
      align = c("l", "c", "r", "r", "r", "r", "r", "r", "r", "r", "r"),
      digits = 2)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.62	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.88	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1

56.7.2 Tabelas Avançadas

```r
# gt: Grammar of Tables
library(gt)

mtcars %>%
  head(5) %>%
  gt() %>%
  tab_header(
    title = "Motor Trend Car Road Tests",
    subtitle = "Primeiras 5 observações"
  ) %>%
  fmt_number(
    columns = c(mpg, disp, hp, drat, wt),
    decimals = 1
  )
```

56.8 1.8 Renderizando Documentos

56.8.1 Via RStudio

- Botão **Knit** (ou Ctrl+Shift+K)
- Escolhe formato automaticamente do YAML

56.8.2 Via Código

```r
# HTML (padrão)
rmarkdown::render("relatorio.Rmd")

# PDF
rmarkdown::render("relatorio.Rmd", output_format = "pdf_document")

# Word
rmarkdown::render("relatorio.Rmd", output_format = "word_document")

# Todos os formatos
rmarkdown::render("relatorio.Rmd", output_format = "all")

# Com parâmetros
rmarkdown::render("relatorio.Rmd",
                  params = list(ano = 2024, regiao = "Sul"))
```

56.9 1.9 Parâmetros

Parâmetros tornam relatórios reutilizáveis.

56.9.1 Definir Parâmetros

```yaml
---
title: "Relatório de Vendas"
output: html_document
params:
  ano: 2024
  regiao: "Sul"
  incluir_graficos: true
---
```

56.9.2 Usar Parâmetros

````
```r
# Acessar parâmetros
ano_selecionado <- params$ano
regiao_selecionada <- params$regiao

# Usar em análise
vendas %>%
  filter(ano == params$ano, regiao == params$regiao) %>%
  summarize(total = sum(vendas))
```
````

````
Análise para o ano de 2024 na região Sul.
(Use código inline com params$ano e params$regiao quando tiver parâmetros definidos)


```r
# Este chunk só executa se incluir_graficos = TRUE
ggplot(dados, aes(x, y)) + geom_point()
```
````

---

57 INTERVALO (20h30 - 20h50)

Aproveite para: - Tomar água/café - Criar seu primeiro RMarkdown - Experimentar com diferentes formatos

---

58 Parte 2: Projeto Final e GitHub Avançado (20h50 - 22h00)

58.1 2.1 Template do Projeto Final

58.1.1 Estrutura

```
projeto-final/
  projeto.Rproj
  README.md
  relatorio.Rmd

  data/
      raw/
          dados.csv
      processed/
           dados_limpos.csv

  scripts/
      01_importar.R
      02_limpar.R
      03_analisar.R

  output/
      figures/
          fig1_distribuicao.png
          fig2_correlacao.png
      tables/
           resumo_estatistico.csv

  docs/
       relatorio.html
```

```
relatorio.pdf
```

58.2 2.2 Exemplo de Projeto Completo

```r
# ============================================================================
# Projeto Final: Análise de Desempenho de Alunos
# Autor: Seu Nome
# Data: 2025-01-XX
# ============================================================================

# Setup ----
library(tidyverse)
library(here)
library(knitr)
library(gt)

# 1. Importar dados ----
dados <- read_csv(here("data", "raw", "alunos.csv"))

# 2. Exploração inicial ----
glimpse(dados)
summary(dados)

# Verificar NAs
dados %>%
  summarize(across(everything(), ~sum(is.na(.))))

# 3. Limpeza ----
dados_limpos <- dados %>%
  # Remover NAs
  drop_na(nota_final) %>%
  # Criar variáveis derivadas
  mutate(
    media = (nota_p1 + nota_p2) / 2,
    status = case_when(
      media >= 7 ~ "Aprovado",
      media >= 5 ~ "Recuperação",
      TRUE ~ "Reprovado"
    ),
    faixa_freq = case_when(
      frequencia >= 90 ~ "Alta",
      frequencia >= 75 ~ "Média",
      TRUE ~ "Baixa"
    )
  ) %>%
  # Filtrar casos válidos
```

```r
  filter(frequencia > 0, nota_p1 > 0, nota_p2 > 0)

# Salvar dados limpos
write_csv(dados_limpos, here("data", "processed", "alunos_limpos.csv"))

# 4. Análise Exploratória ----

## 4.1 Estatísticas descritivas
resumo <- dados_limpos %>%
  summarize(
    n = n(),
    media_geral = mean(media),
    mediana = median(media),
    dp = sd(media),
    aprovados = sum(status == "Aprovado"),
    taxa_aprovacao = mean(status == "Aprovado") * 100
  )

## 4.2 Por curso
por_curso <- dados_limpos %>%
  group_by(curso) %>%
  summarize(
    n = n(),
    media = mean(media),
    taxa_aprovacao = mean(status == "Aprovado") * 100
  ) %>%
  arrange(desc(media))

# Salvar tabela
write_csv(por_curso, here("output", "tables", "resumo_por_curso.csv"))

# 5. Visualizações ----

## 5.1 Distribuição de notas
p1 <- ggplot(dados_limpos, aes(x = media)) +
  geom_histogram(bins = 15, fill = "steelblue", color = "white") +
  geom_vline(xintercept = 7, linetype = "dashed", color = "red") +
  labs(
    title = "Distribuição de Médias Finais",
    subtitle = "Linha vermelha indica nota mínima para aprovação",
    x = "Média Final",
    y = "Frequência"
  ) +
  theme_minimal()

ggsave(here("output", "figures", "fig1_distribuicao.png"),
       p1, width = 8, height = 5, dpi = 300)
```

```r
## 5.2 Média por curso
p2 <- ggplot(por_curso, aes(x = reorder(curso, media), y = media, fill = curso)) +
  geom_col() +
  coord_flip() +
  labs(
    title = "Média de Notas por Curso",
    x = "Curso",
    y = "Média Final"
  ) +
  theme_minimal() +
  theme(legend.position = "none")

ggsave(here("output", "figures", "fig2_por_curso.png"),
       p2, width = 8, height = 5, dpi = 300)

## 5.3 Relação frequência x nota
p3 <- ggplot(dados_limpos, aes(x = frequencia, y = media, color = status)) +
  geom_point(alpha = 0.6, size = 2) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(
    title = "Relação entre Frequência e Desempenho",
    x = "Frequência (%)",
    y = "Média Final",
    color = "Status"
  ) +
  theme_minimal()

ggsave(here("output", "figures", "fig3_frequencia_nota.png"),
       p3, width = 8, height = 5, dpi = 300)

# 6. Conclusões ----
# - Taxa de aprovação geral: XX%
# - Curso com melhor desempenho: XXX
# - Correlação positiva entre frequência e nota
```

58.3 2.3 README.md Profissional

Exemplo de README.md completo:

```
# Análise de Desempenho de Alunos

Projeto final do curso "Introdução à Programação em R com GitHub e IA".

## Descrição

Análise exploratória do desempenho acadêmico de estudantes, investigando
```

```
a relação entre frequência, notas e taxa de aprovação por curso.

## Estrutura do Projeto
```

projeto-final/ data/ # Dados brutos e processados scripts/ # Scripts R de análise output/ # Resultados (figuras e tabelas) docs/ # Relatórios finais

```
## Dados

- **Fonte:** Dataset fictício criado para fins educacionais
- **Observações:** 100 alunos
- **Variáveis:** nome, curso, nota_p1, nota_p2, frequência, bolsista

## Metodologia

1. Importação e validação dos dados
2. Limpeza e tratamento de valores ausentes
3. Criação de variáveis derivadas (média, status)
4. Análise exploratória por curso
5. Visualizações e interpretação

## Principais Resultados

- Taxa de aprovação geral: 78%
- Curso com melhor desempenho: Biologia (média 8.2)
- Correlação positiva entre frequência e nota (r = 0.65)

## Visualizações

![Distribuição de Notas](output/figures/fig1_distribuicao.png)

## Como Reproduzir

```r
# 1. Clone o repositório
git clone https://github.com/usuario/projeto-final.git

# 2. Instale dependências
install.packages(c("tidyverse", "here", "knitr"))

# 3. Execute a análise
source("scripts/01_importar.R")
source("scripts/02_limpar.R")
source("scripts/03_analisar.R")

# 4. Gere o relatório
rmarkdown::render("relatorio.Rmd")
```
```

58.4 Requisitos

- R >= 4.0
- RStudio (recomendado)
- Pacotes: tidyverse, here, knitr

58.5 Autor

Vinícius Silva Junqueira
Email: junqueiravinicius@hotmail.com
Lattes: http://lattes.cnpq.br/4686677580216927
LinkedIn: www.linkedin.com/in/junqueiravinicius

58.6 Licença

MIT License

2.4 .gitignore Adequado

Arquivo `.gitignore` para projetos R:

```gitignore
# R e RStudio
.Rhistory
.RData
.Rproj.user/
*.Rproj.user

# Arquivos temporários do knitr
*_cache/
*_files/
/*.html
/*.pdf
/*.tex

# Dados sensíveis ou muito grandes (descomente se necessário)
# data/raw/*.csv
# data/raw/*.xlsx

# Output temporário
output/temp/

# Sistema
.DS_Store
Thumbs.db
```

```
# LaTeX
*.log
*.aux
*.out
```

58.7 2.5 Git e GitHub Avançado

58.7.1 Commits Significativos

```
# RUIM
git commit -m "update"
git commit -m "fix"

# BOM
git commit -m "Adiciona análise exploratória de frequência"
git commit -m "Corrige cálculo de média ponderada"
git commit -m "Atualiza visualizações com tema consistente"
```

58.7.2 Branches Básicas

```
# Criar branch para nova feature
git checkout -b analise-correlacao

# Fazer mudanças e commits
git add scripts/04_correlacao.R
git commit -m "Adiciona análise de correlação"

# Voltar para main
git checkout main

# Merge (integrar mudanças)
git merge analise-correlacao

# Deletar branch
git branch -d analise-correlacao
```

58.7.3 Tags (Versões)

```
# Criar tag
git tag -a v1.0 -m "Versão final do projeto"

# Push com tags
git push origin main --tags

# Listar tags
git tag
```

58.8 2.6 Usando IA para Documentação

58.8.1 Prompts Eficientes

```
# Para ChatGPT/Claude:

# "Revise este README.md e sugira melhorias:
# [cole seu README]"

# "Crie um README profissional para este projeto de análise de dados.
# O projeto analisa [descreva]. Principais resultados: [liste]."

# "Escreva comentários para este código de forma clara:
# [cole código]"

# "Gere uma descrição de commit apropriada para estas mudanças:
# - Adicionei análise de correlação
# - Criei 3 novas visualizações
# - Corrigi bug no cálculo de média"
```

---

59 Exercícios Práticos

59.1 Exercício 1: Primeiro RMarkdown

```
# a) Crie novo RMarkdown: File > New File > R Markdown


# b) Modifique o YAML:
#    - Adicione seu nome
#    - Mude para theme: flatly
#    - Adicione toc: true


# c) Adicione novo chunk com análise de mtcars


# d) Use código inline para mostrar número de observações


# e) Gere HTML e visualize
```

59.2 Exercício 2: Relatório com Dados Reais

```
# Crie relatório RMarkdown que:

# a) Carrega dataset (use iris ou mtcars)


# b) Mostra resumo estatístico em tabela formatada (kable)


# c) Cria 3 visualizações diferentes


# d) Interpreta resultados com texto entre os chunks


# e) Gera PDF e HTML
```

59.3 Exercício 3: Projeto Organizado

```
# Organize um projeto completo:

# a) Crie estrutura de pastas (data/, scripts/, output/)


# b) Coloque dados em data/raw/


# c) Crie script de limpeza em scripts/


# d) Crie script de visualização em scripts/


# e) Gere relatório final em docs/
```

59.4 Exercício 4: README Profissional

```
# Para seu projeto:

# a) Crie README.md com:
#    - Título e descrição
#    - Estrutura de arquivos
#    - Como reproduzir
#    - Principais resultados
#    - Suas informações de contato
```

```
# b) Adicione badge de licença


# c) Inclua pelo menos uma visualização


# d) Use IA para melhorar o texto
```

59.5 Exercício 5: Projeto Final Completo

```
# Projeto final integrado:

# 1. Escolha tema (use seus dados ou dataset público)


# 2. Crie estrutura profissional de projeto


# 3. Faça análise exploratória completa


# 4. Crie pelo menos 5 visualizações


# 5. Escreva relatório RMarkdown com:
#    - Introdução
#    - Metodologia
#    - Resultados
#    - Conclusões


# 6. Configure README.md e .gitignore


# 7. Faça commits organizados


# 8. Push para GitHub


# 9. Use IA para revisar e melhorar


# 10. Compartilhe o link!
```

---

60 Apresentações dos Projetos

60.1 Formato (3-5 minutos cada)

1. **Contexto:** Qual pergunta você tentou responder?
2. **Dados:** Que dados usou?
3. **Análise:** O que fez?
4. **Resultados:** O que descobriu?
5. **Aprendizados:** O que foi mais desafiador?

60.2 Dicas para Apresentar

- Mostre visualizações principais
- Explique insights interessantes
- Mencione dificuldades superadas
- Compartilhe link do GitHub
- Aceite feedback construtivo

---

61 Recursos para Continuar Aprendendo

61.1 Livros Online (Gratuitos)

- R for Data Science (2e) - Essencial
- Advanced R - Nível avançado
- R Graphics Cookbook - Visualizações
- R Markdown Cookbook
- Happy Git with R

61.2 Comunidades

- Posit Community
- Stack Overflow [r]
- R-Ladies - Capítulos em várias cidades
- Reddit r/rstats
- RWeekly - Newsletter semanal

61.3 Prática Contínua

- TidyTuesday - Desafios semanais
- Kaggle - Competições e datasets
- DataCamp - Cursos interativos (pago)
- Coursera - Especializações em R

61.4 Datasets Públicos

- data.gov - Dados governamentais (EUA)
- Brasil.io - Dados públicos brasileiros
- IBGE - Instituto Brasileiro de Geografia

- DataSUS - Dados de saúde
- UCI Machine Learning - Datasets variados

61.5 Cheat Sheets

- RStudio: Help → Cheat Sheets
- Posit Cheat Sheets

61.6 Canais YouTube (em Português)

- Curso-R
- Fernanda Peres
- Análise Macro

---

62 Certificado de Conclusão

62.1 Você completou o curso!

Conteúdo dominado:

- **Dia 1:** R, RStudio, Git, fundamentos de programação
- **Dia 2:** Lógica, funções, boas práticas, debugging
- **Dia 3:** Manipulação de dados com tidyverse
- **Dia 4:** I/O de dados e visualização com ggplot2
- **Dia 5:** RMarkdown e projetos reprodutíveis

62.2 Próximos Passos

1. **Pratique regularmente** - 15 min/dia é melhor que 2h/semana
2. **Participe de comunidades** - Tire dúvidas, ajude outros
3. **Trabalhe em projetos reais** - Use seus próprios dados
4. **Contribua no GitHub** - Mostre seu portfólio
5. **Continue aprendendo** - R é uma jornada sem fim!

---

63 Feedback do Curso

Por favor, dedique alguns minutos para avaliar o curso:

O que funcionou bem? - Conteúdo claro e organizado? - Exercícios adequados? - Ritmo apropriado? - Material de apoio suficiente?

O que pode melhorar? - Tópicos que faltaram? - Partes confusas? - Mais/menos exercícios? - Sugestões gerais?

63.1 Onde Deixar Feedback

- Issues no GitHub do curso

- Email: junqueiravinicius@hotmail.com
- Formulário online: [link se disponível]

Seu feedback é essencial para melhorar o curso!

---

64 Agradecimentos

64.1 Obrigado por Participar!

Espero que este curso tenha sido útil e inspirador. Lembre-se:

- **Erros são normais** - São parte do aprendizado
- **Google é seu amigo** - Ninguém sabe tudo de cor
- **Comunidade ajuda** - Não hesite em pedir ajuda
- **Prática é fundamental** - Continue codando!
- **IA é ferramenta** - Use com sabedoria

64.2 Mantenha Contato

- **Email:** junqueiravinicius@hotmail.com
- **GitHub:** github.com/viniciusjunqueira
- **Lattes:** lattes.cnpq.br/4686677580216927
- **LinkedIn:** linkedin.com/in/junqueiravinicius

64.3 Grupo de Ex-Alunos

[Se aplicável] Entre no grupo para: - Tirar dúvidas - Compartilhar projetos - Networking - Vagas e oportunidades

---

65 Commit Final

Versione seu projeto final:

```
git add .
git commit -m "Projeto final: análise completa e relatório"
git push

# Criar release
git tag -a v1.0 -m "Versão final do curso"
git push origin v1.0
```

---

66 Para Casa (Forever!)

1. **Revise** todo o material do curso
2. **Complete** seu projeto final se não terminou

3. **Compartilhe** no GitHub
4. **Junte-se** a comunidades R
5. **Pratique** com TidyTuesday
6. **Leia** R for Data Science
7. **Aplique** R em seu trabalho/pesquisa
8. **Ensine** outros - melhor forma de aprender!

---

67 Recursos Finais

67.1 Template de Projeto

```
meu-projeto-r/
   README.md
   LICENSE
   .gitignore
   meu-projeto.Rproj
   data/
       raw/
       processed/
   scripts/
       01_import.R
       02_clean.R
       03_analyze.R
       04_visualize.R
   output/
       figures/
       tables/
   docs/
       relatorio.Rmd
   tests/
        test_functions.R
```

67.2 Script de Setup Automático

```r
# Função para criar estrutura de projeto
criar_projeto <- function(nome) {
  # Criar pastas
  dir.create(nome)
  setwd(nome)

  pastas <- c("data/raw", "data/processed", "scripts",
              "output/figures", "output/tables", "docs", "tests")

  for (pasta in pastas) {
    dir.create(pasta, recursive = TRUE)
  }
```

```r
  # Criar arquivos
  file.create("README.md")
  file.create(".gitignore")

  # .gitignore básico
  writeLines(c(
    ".Rhistory",
    ".RData",
    ".Rproj.user/",
    "*.html",
    "output/temp/"
  ), ".gitignore")

  # README template
  writeLines(c(
    paste("#", nome),
    "",
    "## Descrição",
    "",
    "## Como usar",
    "",
    "## Autor"
  ), "README.md")

  message("Projeto criado com sucesso!")
}

# Usar
criar_projeto("meu-projeto-incrivel")
```

---

68 Conclusão

68.1 Parabéns por Completar o Curso!

Você agora tem as ferramentas fundamentais para:

- Analisar dados de forma reprodutível
- Criar visualizações profissionais
- Colaborar usando Git/GitHub
- Documentar análises com RMarkdown
- Usar IA produtivamente
- Continuar aprendendo autonomamente

68.2 A Jornada Continua

R é uma linguagem em constante evolução. Continue:

- Explorando novos pacotes
- Aprendendo novas técnicas
- Compartilhando conhecimento
- Construindo seu portfólio
- Contribuindo para a comunidade

68.3 Boa Sorte!

Sucesso em suas análises e projetos futuros!

Nos vemos na comunidade R!

Última atualização: 2025-10-07
Contato: junqueiravinicius@hotmail.com
Repositório: https://github.com/viniciusjunqueira/curso-r-github-ia
Lattes: http://lattes.cnpq.br/4686677580216927

---

69 FIM DO CURSO

Obrigado por sua dedicação e participação!

70 Conclus o e Pr ximos Passos

70.1 O que voc aprendeu

Ao longo deste curso, voc desenvolveu habilidades em:

1. **Fundamentos de R:** tipos de dados, estruturas, manipula o b sica
2. **Programa o:** l gica, fun es, debugging, boas pr ticas
3. **Manipula o de dados:** tidyverse, transforma es, limpeza
4. **Visualiza o:** ggplot2, gr ficos profissionais, customiza o
5. **Reprodutibilidade:** RMarkdown, projetos organizados, Git/GitHub
6. **IA Assistida:** uso estrat gico de ChatGPT e Claude

70.2 Recursos para Continuar Aprendendo

70.2.1 Livros Online (Gratuitos)

- **R for Data Science (2e):** https://r4ds.hadley.nz/
- **Advanced R:** https://adv-r.hadley.nz/
- **R Graphics Cookbook:** https://r-graphics.org/
- **Happy Git with R:** https://happygitwithr.com/

70.2.2 Comunidades

- **Posit Community:** https://community.rstudio.com/
- **Stack Overflow:** tag [r]
- **R-Ladies:** cap tulos locais e globais
- **Reddit:** r/rstats, r/Rlanguage

70.2.3 Pr tica Cont nua

- **TidyTuesday:** https://github.com/rfordatascience/tidytuesday
- **Kaggle:** https://www.kaggle.com/datasets
- **Data.gov:** https://data.gov/ (dados p blicos)
- **Brasil.io:** https://brasil.io/datasets/

70.2.4 Cursos Avan ados

- Posit Academy
- DataCamp
- Coursera (Johns Hopkins - Data Science Specialization)
- edX

70.3 Certificado de Conclus o

Parab ns por concluir o curso! Voc agora possui as habilidades fundamentais para:

- Conduzir an lises de dados profissionais
- Criar visualiza es impactantes
- Trabalhar de forma reprodut vel
- Colaborar usando Git/GitHub
- Continuar aprendendo autonomamente

70.4 Mantenha Contato

Instrutor: Vin cius Silva Junqueira
Email: seu-email@example.com
GitHub: https://github.com/seu-usuario
LinkedIn: [seu perfil]

70.4.1 Suporte P s-Curso

- D vidas via Issues no reposit rio do curso
- Sess o de follow-up ap s 2 semanas (se aplic vel)
- Comunidade de ex-alunos (se aplic vel)

** ltima atualiza o:** 2025-10-07
Vers o: 1.0