Introdução Programação em R com GitHub, ChatGPT e Claude

Vinícius Silva Junqueira

2025-10-07

Sumário

1 Dia 2 — Lógica, Funções e Introdução ao Tidyverse

Objetivos do dia

- Dominar operadores lógicos/relacionais e condicionais (`if`, `ifelse`, `case_when`).
- Entender loops vs. vetorização e criar **funções próprias**.
- Aplicar um **pipeline básico** com `dplyr` e introduzir **datas** com `lubridate`.
- Registrar o aprendizado com um commit no seu fork no GitHub.

Tempo previsto 19h00–22h00 (intervalo 20h30–20h50)

1.1 1. Operadores e Condicionais (30 min)

1.1.1 1.1 Operadores lógicos e relacionais

```
# Lógicos: & | ! xor()
TRUE & FALSE
TRUE | FALSE
!TRUE
xor(TRUE, FALSE)
```

```r
# Relacionais: == != > < >= <=
3 == 3
5 != 2
5 > 2; 1 < 0
2 >= 2; 3 <= 10

# %in% (teste de pertinência)
2 %in% c(1, 2, 3)
"Adelie" %in% c("Chinstrap", "Gentoo")
```

1.1.2 1.2 Condicionais: `if`, `else`, `ifelse`, `case_when`

```r
# if/else (escala escalar)
x <- 18
if (x >= 18) {
  status <- "maior_de_idade"
} else {
  status <- "menor_de_idade"
}
status

# ifelse() (vetorizado)
notas <- c(5.9, 7.5, 9.2, 6.0)
resultado <- ifelse(notas >= 7, "Aprovado", "Recuperação")
resultado

# case_when() (multiplas regras)
# require(dplyr)
# dplyr::case_when avalia em ordem
library(dplyr)
faixa <- case_when(
  notas >= 9           ~ "Excelente",
  notas >= 7 & notas < 9 ~ "Bom",
  notas >= 5 & notas < 7 ~ "Regular",
  TRUE                  ~ "Insuficiente"
)
faixa
```

> **Dica didática**: usar `ifelse` quando quiser **vetorizar**; `case_when` quando houver **várias regras**.

1.2 2. Loops, Vetorização e Funções (25 min)

1.2.1 2.1 Loops vs. operações vetorizadas

```r
valores <- 1:5

# Loop for (didático)
soma <- 0
for (v in valores) {
  soma <- soma + v
}
soma

# Vetorizado (mais idiomático em R)
sum(valores)
```

1.2.2 2.2 Sua primeira função: IMC

```r
# Fórmula: IMC = peso(kg) / altura(m)^2
imc <- function(peso, altura) {
  if (any(altura <= 0)) stop("Altura deve ser > 0")
  peso / (altura ^ 2)
}

imc(c(70, 80), c(1.70, 1.80))

# Classificando IMC usando case_when()
classificar_imc <- function(imc) {
  dplyr::case_when(
    imc < 18.5              ~ "Abaixo do peso",
    imc >= 18.5 & imc < 25  ~ "Normal",
    imc >= 25   & imc < 30  ~ "Sobrepeso",
    imc >= 30               ~ "Obesidade"
  )
}

val <- imc(80, 1.75)
classificar_imc(val)
```

> **Boas práticas**: funções **nomeiam intenções**, validam entradas e retornam **um** resultado claro.

1.3 3. Introdução ao Tidyverse (45 min)

Vamos aplicar `dplyr` no dataset **palmerpenguins** e criar um pequeno pipeline.

```r
library(dplyr)
library(palmerpenguins)

# Remover linhas com NAs nas colunas essenciais
peng <- penguins |>
```

```r
  filter(!is.na(species),
         !is.na(bill_length_mm),
         !is.na(bill_depth_mm),
         !is.na(flipper_length_mm),
         !is.na(body_mass_g))

# Selecionar só o que precisamos
peng_sel <- peng |>
  select(species, island, bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g)

# Criar nova variável (razão do bico) e reordenar
peng_feat <- peng_sel |>
  mutate(raz_bico = bill_length_mm / bill_depth_mm) |>
  arrange(species, desc(raz_bico))

# Resumo por espécie
resumo <- peng_feat |>
  group_by(species) |>
  summarize(
    n = n(),
    media_flipper = mean(flipper_length_mm),
    sd_flipper    = sd(flipper_length_mm),
    media_massa   = mean(body_mass_g)
  )
resumo
```

1.3.1 3.1 Pipe: %>% vs. |>

```r
# Ambos funcionam; escolha um padrão para a turma.
# Exemplo com |> (pipe nativo do R >= 4.1):
penguins |>
  tidyr::drop_na(bill_length_mm) |>
  dplyr::summarize(media = mean(bill_length_mm))
```

1.4 4. Datas com `lubridate` (10 min)

Datas aparecem em **quase todos** os projetos. Vamos ilustrar rapidamente.

```r
library(lubridate)

# Criação e parsing
ymd("2025-11-18")
dmy("18/11/2025")
mdy("11-18-2025")

# Componentes
```

```r
hoje <- today()
ano(hoje); mes(hoje); wday(hoje, label = TRUE, abbr = FALSE)

# Operações simples
hoje + days(14)
interval(ymd("2025-11-01"), ymd("2025-11-18"))
```

Integrando no pipeline: quando houver colunas de data, transforme-as e derive **mês/ano** para agregações.

1.5 5. Exercícios Práticos (20–25 min)

Dataset: `palmerpenguins::penguins`

1.5.1 Exercício 1 — Condicionais

1. Crie um vetor de 8 notas qualquer.
2. Classifique com `ifelse` como **Aprovado/Recuperação** (corte em 7).
3. Depois, crie uma classificação mais rica usando `case_when` com 4 faixas.

```r
# Seu código aqui
```

1.5.2 Exercício 2 — Funções

1. Escreva uma função `zscore(x)` que centraliza e escala (média 0, desvio 1).
2. Aplique em `bill_length_mm` **removendo NAs** antes.
3. Faça um segundo argumento opcional `na_rm = TRUE` dentro da função.

```r
# Seu código aqui
```

1.5.3 Exercício 3 — Pipeline `dplyr`

1. Crie `peng3` filtrando linhas completas nas 4 medidas principais.
2. Calcule, por espécie, média e desvio da nadadeira (`flipper_length_mm`).
3. Ordene do maior para o menor e mostre as 5 primeiras linhas.

```r
# Seu código aqui
```

1.5.4 Exercício 4 — Datas com `lubridate`

1. Crie um vetor com 5 datas em formato “dd/mm/aaaa”.

2. Converta com `dmy()` e extraia `month()` (com rótulo).

3. Some 30 dias à primeira data e compute o intervalo até a última.

```
# Seu código aqui
```

1.6 6. Boas Práticas e Debugging (20 min)

- Use **nomes descritivos** em `snake_case`.
- Comente o **porquê** (não só o que) no código.
- Valide entradas em funções (`stop()` para erros previsíveis).
- Leia mensagens de erro **de baixo para cima** (stack trace).
- Mantenha scripts curtos e reutilizáveis.

1.6.1 Ferramentas úteis

```
# message(), warning(), stop() para sinalizar eventos
# browser() para inspecionar dentro de uma função (quando eval=TRUE)
# traceback() após um erro
```

> **IA como apoio (responsável):** use ChatGPT/Claude para **explicar erros** e sugerir melhorias, mas sempre **entenda e teste** o código.

1.7 7. Commit do Dia

1. Salve como `scripts/02_logica_funcoes.R` ou `materiais/dia2_logica_funcoes.Rmd` (este arquivo).

2. No **Terminal do RStudio**:

```
git add scripts/02_logica_funcoes.R
git commit -m "Dia 2: lógica, funções e tidyverse (com lubridate)"
git push origin main
```

> Lembre-se: você está trabalhando **no SEU fork**. O repositório original permanece protegido.

1.8 8. Referências rápidas

- **`dplyr` cheatsheet**: https://posit.co/resources/cheatsheets/
- **R for Data Science (2e)**: https://r4ds.hadley.nz/
- **Happy Git with R**: https://happygitwithr.com/
- **palmerpenguins**: https://allisonhorst.github.io/palmerpenguins/