Introdução Programação em R com GitHub, ChatGPT e Claude

Vinícius Silva Junqueira

2025-10-07

Sumário

1 Informações Gerais do Curso

Carga horária total: 16 horas
Datas: 17/11, 18/11, 24/11 e 25/11
Horário: 19h00 às 22h00 (com intervalo de 20 minutos às 20h30)
Tempo líquido por dia: 2h40 de conteúdo efetivo
Público-alvo: Iniciantes de diversas áreas (ciências agrárias, saúde, economia, biologia, ciências sociais)
Material: Repositório GitHub com datasets, scripts e exercícios
Repositório original: https://github.com/viniciusjunqueira/curso-r-github-ia
Sistemas Operacionais: Windows, macOS e Linux (curso compatível com todos)

1.1 Importante: Workflow com Fork

Cada aluno deve fazer um FORK do repositório original para trabalhar em sua própria cópia!

O que você vai fazer no Dia 1: 1. Criar conta no GitHub (se não tiver) 2. Configurar Git no seu computador 3. Autenticar no GitHub 4. **FAZER FORK** → Criar sua cópia pessoal do repositório 5. **CLONAR SEU FORK** → Baixar para seu computador 6. Trabalhar e fazer commits 7. **PUSH PARA SEU FORK** → Suas mudanças vão para SUA cópia

Entenda a diferença:

```
ERRADO - Clonar diretamente o original:
  viniciusjunqueira/repo → [CLONE] → Seu computador
  Resultado: Você NÃO conseguirá fazer push!

CORRETO - Fazer fork primeiro:
  viniciusjunqueira/repo → [FORK] → SEU-USUARIO/repo
                                          ↓
                                      [CLONE]
                                          ↓
                                    Seu computador
  Resultado: Você PODE fazer push para SEU fork!
```

Vantagens do fork: - Você tem controle total sobre sua cópia - Pode fazer commits e push sem restrições - O repositório original permanece protegido - Aprende workflow real de contribuição open source - Se errar, não afeta ninguém além de você

Instruções detalhadas sobre como fazer o fork estão na seção "Fork do Repositório do Curso" no Dia 1.

1.2 Pré-requisitos Técnicos

1.2.1 Instalações Obrigatórias (para todos os sistemas)

1.2.1.1 1. R (versão 4.3 ou superior)

- **Download:** https://cran.r-project.org/
- **Windows:** Executar instalador `.exe`
- **macOS:** Instalar arquivo `.pkg` (verificar se é Apple Silicon ou Intel)
- **Linux (Ubuntu/Debian):**

```
sudo apt update
sudo apt install r-base r-base-dev
```

1.2.1.2 2. RStudio Desktop (versão 2023.09 ou superior)

- **Download:** https://posit.co/download/rstudio-desktop/
- **Todos os sistemas:** Instalar o arquivo apropriado para seu SO
- **Linux:** Também disponível via `.deb` ou `.rpm`

1.2.1.3 3. Git

- **Windows:** https://git-scm.com/download/win (inclui Git Bash)
- **macOS:**
  - Vem pré-instalado (verificar com `git --version` no Terminal)
  - Ou instalar via: https://git-scm.com/download/mac
- **Linux:**

```
sudo apt install git  # Ubuntu/Debian
sudo yum install git  # CentOS/Fedora
```

1.2.1.4 4. Conta no GitHub

- **Todos os sistemas:** https://github.com/signup
- Gratuita e necessária para o curso

1.3 Estrutura Pedagógica

- **Multiplataforma:** Todo conteúdo funciona em Windows, macOS e Linux
- **RStudio como IDE padrão:** Garante experiência consistente entre sistemas
- **Teoria + Prática:** Cada conceito é seguido de exercício prático
- **Aprendizado Incremental:** Commits no GitHub ao final de cada dia
- **Casos Multidisciplinares:** Exemplos de várias áreas do conhecimento
- **IA como Ferramenta:** Uso estratégico de ChatGPT e Claude
- **Caminhos relativos:** Uso de `here::here()` para portabilidade total

---

1.4 Pacotes R Necessários

1.4.1 Instalação no Dia 1 (núcleo do curso)

```r
# Núcleo
install.packages(c(
  "tidyverse", "here", "janitor", "skimr",
  "readxl", "writexl", "rmarkdown", "knitr",
  "lubridate", "scales", "patchwork", "broom"
))

# Datasets didáticos
install.packages("palmerpenguins")
```

1.4.2 Instalação opcional (Dia 4 – IA e APIs)

```r
# Opcionais (não essenciais ao núcleo iniciante)
install.packages(c("gptstudio", "chattr", "httr2"))
```

Nota: IA e httr2 permanecem como extras, para quem tiver API keys e interesse em explorar.

---

2 Dia 1: 17/11 (Segunda) - Fundamentos e Ambiente de Trabalho

2.1 19h00 - 20h30 | Ambientação e Setup Completo

2.1.1 Apresentação do Curso (20min)

- Objetivos e metodologia
- Por que R, GitHub e IA?
- Panorama das áreas de aplicação
- Apresentação dos participantes e expectativas
- **Verificação dos sistemas:** todos com R, RStudio e Git instalados?

2.1.2 Configuração do Ambiente (50min)

Visão Geral do Processo: Antes de começar, entenda o fluxo completo que vamos seguir:

1. **Verificar instalações** (R, RStudio, Git)
2. **Configurar Git local** (nome e email)
3. **Autenticar no GitHub** (PAT, GitHub Desktop ou SSH)
4. **FAZER FORK** do repositório (criar SUA cópia)
5. **Clonar SEU fork** para seu computador
6. **Trabalhar nos seus arquivos** localmente
7. **Fazer commit e push** para SEU fork

Por que este fluxo? - Você trabalha em SUA cópia independente - Não interfere no repositório original do instrutor - Aprende o workflow real usado em projetos open source

---

```r
# No Console do RStudio (funciona em todos os sistemas)
R.version.string          # Verificar versão do R
RStudio.Version()$version # Verificar versão do RStudio
system("git --version")   # Verificar Git (funciona em todos os SO)
```

2.1.2.1 Verificação das Instalações

2.1.2.2 Configuração Git Local (multiplataforma) **No Terminal do RStudio** (recomendado - funciona em todos os sistemas):

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

Verificar configuração:

```
git config --global --list
```

Nota: O RStudio Terminal funciona igual no Windows, macOS e Linux!

2.1.2.3 Autenticação GitHub (3 métodos - escolha o mais fácil) Método 1: Personal Access Token (PAT) - Recomendado

```r
# No Console do RStudio (todos os sistemas)
install.packages("usethis")
install.packages("gitcreds")

usethis::create_github_token()  # Abre navegador para criar token
gitcreds::gitcreds_set()        # Cole o token quando solicitado
```

Método 2: GitHub Desktop (Interface Gráfica) - Download: https://desktop.github.com/ - Disponível para Windows e macOS - Autenticação automática via navegador

Método 3: SSH Keys (Avançado - Opcional) - Configuração via `ssh-keygen` (todos os sistemas) - Instruções: https://happygitwithr.com/ssh-keys.html

---

2.1.3 Fork do Repositório do Curso (15min)

PASSO OBRIGATÓRIO: Cada aluno deve criar sua própria cópia (fork) do repositório!

2.1.3.1 Por que fazer Fork?

- Você terá controle total sobre sua cópia pessoal
- Poderá fazer commits e push sem restrições
- O repositório original do instrutor permanece protegido
- Aprende o workflow real usado em projetos open source

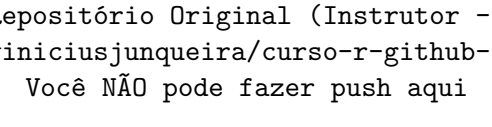
2.1.3.2 Passo a Passo: Como Fazer o Fork 1. Acessar o repositório original (3min) - Abra seu navegador - Acesse: https://github.com/viniciusjunqueira/curso-r-github-ia - Faça login na sua conta GitHub (se ainda não estiver logado)

2. Criar o Fork (2min) - No canto superior direito da página, clique no botão **"Fork"** - Uma tela aparecerá perguntando onde criar o fork - **Owner:** Selecione sua conta (seu nome de usuário) - **Repository name:** Mantenha `curso-r-github-ia` (pode deixar igual) - **Description:** (Opcional) Você pode adicionar uma descrição - **Desmarque** a opção "Copy the main branch only" se quiser todas as branches - Clique em **"Create fork"** (botão verde)

3. Aguardar a criação (1min) - O GitHub irá copiar todo o repositório para sua conta - Quando terminar, você será redirecionado automaticamente - Agora você tem: `https://github.com/SEU-USUARIO/curso-r-github-ia`

4. Verificar que é seu fork (1min) - No topo da página, você verá: `SEU-USUARIO / curso-r-github-ia forked from viniciusjunqueira/curso-r-github-ia` - Isso confirma que é SUA cópia pessoal!

2.1.3.3 Diagrama do Workflow



```
                    FORK (você cria uma cópia)
                   ↓

  Seu Fork no GitHub (Sua conta - VOCÊ CONTROLA)
  SEU-USUARIO/curso-r-github-ia
    Você PODE fazer push aqui


                    CLONE (você baixa localmente)
                   ↓

  Seu Computador Local
  ~/Documents/curso-r-github-ia/
    Aqui você edita arquivos e faz commits


                    PUSH (você envia mudanças)
                   ↓
          Volta para seu Fork no GitHub
```

2.1.3.4 Notas Importantes

- **SEMPRE clone SEU fork**, não o repositório original
- A URL do seu fork é: `https://github.com/SEU-USUARIO/curso-r-github-ia`
- Substitua `SEU-USUARIO` pelo seu nome de usuário do GitHub
- Se clonar o repo errado, você não conseguirá fazer push!

2.1.4 Clonagem do SEU Fork (20min)

IMPORTANTE: Clone SEU fork, não o repositório original!

2.1.4.1 Opção 1: Via RStudio (Recomendado - Funciona em Todos)

1. File → New Project → Version Control → Git
2. **Repository URL:** `https://github.com/SEU-USUARIO/curso-r-github-ia`
   - **Atenção:** Use **SEU-USUARIO**, não `viniciusjunqueira`!
   - Exemplo: Se seu usuário é `maria123`, use: `https://github.com/maria123/curso-r-github-ia`
3. **Project directory name:** Deixe como `curso-r-github-ia`
4. **Create project as subdirectory of:** Escolha pasta local (ex: `~/Documents`)
5. Clique em **"Create Project"**
6. Aguarde o download - pode demorar alguns minutos

```
# Substitua SEU-USUARIO pelo seu nome de usuário do GitHub!
git clone https://github.com/SEU-USUARIO/curso-r-github-ia.git
```

```
cd curso-r-github-ia
```

2.1.4.2 Opção 2: Via Terminal do RStudio

Depois: File → Open Project → Selecionar `.Rproj`

2.1.4.3 Opção 3: GitHub Desktop (Windows/macOS)

1. Abra o GitHub Desktop
2. File → Clone Repository
3. Na aba "GitHub.com", procure por `curso-r-github-ia`
4. **Importante:** Selecione o que está em **SUA conta**, não do instrutor!
5. Escolha pasta local
6. Clone
7. Depois: Repository → Open in RStudio

```
# No Terminal do RStudio, verificar repositórios remotos
git remote -v

#  Deve aparecer SEU usuário:
# origin  https://github.com/SEU-USUARIO/curso-r-github-ia.git (fetch)
# origin  https://github.com/SEU-USUARIO/curso-r-github-ia.git (push)

#  Se aparecer viniciusjunqueira, você clonou o errado!
# Veja solução na seção Troubleshooting
```

2.1.4.4 Verificar se Clonou o Fork Correto

2.1.5 Estrutura de Pastas e Portabilidade

```
curso-r-github-ia/
   curso-r-github-ia.Rproj  # SEMPRE use projetos!
   data/
      raw/                  # Dados originais
      processed/            # Dados limpos
   scripts/                  # Scripts R
   output/                   # Resultados
      figures/
      tables/
   docs/                     # Documentação
```

IMPORTANTE - Caminhos Relativos (Portabilidade):

```
#  RUIM - Específico de sistema
dados <- read.csv("C:/Users/Nome/Documents/dados.csv")  # Só Windows
dados <- read.csv("/Users/nome/Documents/dados.csv")    # Só macOS

#  BOM - Funciona em todos os sistemas
library(here)
dados <- read.csv(here("data", "raw", "dados.csv"))
```

```
# O here::here() constrói caminhos corretos automaticamente!
```

2.2 20h30 - 20h50 | INTERVALO

2.3 20h50 - 22h00 | Fundamentos do R

2.3.1 Objetos e Estruturas de Dados (40min)

- Vetores: numérico, lógico, string
- Listas e data.frames
- Fatores: variáveis categóricas
- Funções básicas: `c()`, `length()`, `class()`, `typeof()`

Nota: Todo código R funciona identicamente em Windows, macOS e Linux!

2.3.2 Exploração Inicial de Dados (30min)

- `str()`, `head()`, `tail()`, `names()`
- `dplyr::glimpse()` (visão moderna)
- `summary()` para estatísticas descritivas
- **Indexação:** por posição `[1]`, por nome `$coluna`, por condição

2.3.3 Prática Guiada (20min)

- Criar vetores e data.frames
- Manipular objetos básicos
- **Primeiro script:** salvar como `01_fundamentos.R`
- **Salvando com encoding correto (importante!):**
  - File → Save with Encoding → UTF-8 (para acentos funcionarem)

2.3.4 Primeiro Commit (multiplataforma)

No Terminal do RStudio:

```
# Adicionar arquivo ao stage
git add scripts/01_fundamentos.R

# Criar commit com mensagem
git commit -m "Fundamentos do R - Dia 1"

# Enviar para SEU fork no GitHub
git push origin main
# Nota: "origin" aponta para SEU fork
```

Alternativa - Interface Gráfica do RStudio: 1. Vá na aba **"Git"** (painel superior direito) 2. Marque o checkbox dos arquivos que quer commitar 3. Clique em **"Commit"** 4. Escreva

mensagem: `Fundamentos do R - Dia 1` 5. Clique em **"Commit"** 6. Clique em **"Push"** (seta verde apontando para cima)

Verificação: - Acesse `https://github.com/SEU-USUARIO/curso-r-github-ia` - Você deve ver seu commit lá! - O repositório original do instrutor **NÃO** será afetado

2.4 Para Casa

- Revisar conceitos no material do repositório
- Experimentar com seus próprios dados (se tiver)
- **Lembrete:** Todos os commits vão para SEU fork, não afetam o repositório original

---

2.5 Checklist - Verifique se tudo está correto

Antes de terminar o Dia 1, confirme cada item:

- ☐ **Git configurado:** `git config --global --list` mostra meu nome e email
- ☐ **GitHub autenticado:** Consegui criar PAT e configurar com `gitcreds`
- ☐ **Fork criado:** Existe `https://github.com/MEU-USUARIO/curso-r-github-ia`
- ☐ **Repositório clonado:** Tenho pasta `curso-r-github-ia` no meu computador
- ☐ **Projeto RStudio aberto:** Vejo `.Rproj` na pasta
- ☐ **Remote correto:** `git remote -v` mostra MEU usuário (não viniciusjunqueira)
- ☐ **Primeiro commit feito:** Script `01_fundamentos.R` existe no GitHub
- ☐ **Push funcionou:** Consigo ver minhas mudanças no meu fork do GitHub
- ☐ **Pacotes instalados:** tidyverse, here, janitor, skimr, readxl, writexl, rmarkdown, knitr, lubridate, scales, patchwork, broom, palmerpenguins

Se algum item não estiver marcado, peça ajuda ao instrutor!

2.6 (Opcional) Sincronizar com Repositório Original

Se o instrutor fizer atualizações no repositório original durante o curso, você pode sincronizar seu fork:

Configurar uma vez (adicionar "upstream"):

```
# Adicionar repositório original como "upstream"
git remote add upstream https://github.com/viniciusjunqueira/curso-r-github-ia.git

# Verificar configuração
git remote -v
# Deve mostrar:
# origin    (seu fork)
# upstream  (original do instrutor)
```

Quando o instrutor avisar de atualizações, sincronize:

```
# 1. Buscar atualizações do original
git fetch upstream
```

```
# 2. Ir para sua branch principal
git checkout main

# 3. Mesclar atualizações na sua branch
git merge upstream/main

# 4. Enviar para seu fork no GitHub
git push origin main
```

Alternativa via Interface do GitHub: 1. Acesse seu fork: `https://github.com/SEU-USUARIO/curso-r-githu`
2. Se houver atualizações, aparecerá: "This branch is X commits behind viniciusjunqueira:main"
3. Clique em **"Sync fork"** → **"Update branch"** 4. No RStudio: `git pull origin main` para baixar as atualizações

---

3 Dia 2: 18/11 (Terça) - Lógica, Funções e Introdução ao Tidyverse

3.1 19h00 - 20h30 | Programação em R

3.1.1 Operadores e Condicionais (30min)

- **Operadores lógicos:** `&`, `|`, `!`, `%in%`
- **Operadores relacionais:** `==`, `!=`, `>`, `<`, `>=`, `<=`
- **Condicionais:**
  - `if` e `else` (estrutura completa)
  - `ifelse()` vetorizado
  - `dplyr::case_when()` para múltiplas condições

3.1.2 Loops e Funções (25min)

- Loop `for`: sintaxe e casos de uso
- Vetorização vs. loops (eficiência)
- **Funções customizadas:**
  - Sintaxe: `function(argumentos) { corpo }`
  - Argumentos e retorno
  - **Exemplo prático:** função para calcular IMC

3.1.3 Boas Práticas e Debugging (20min)

- **Naming conventions:** `snake_case`
- **Comentários eficientes**
- **Lendo mensagens de erro**
- **IA como assistente:** ChatGPT vs Claude (ética: sempre entender o código)

3.1.4 IA Assistida (5min)

- Depurar scripts com erros
- Usar ChatGPT/Claude para explicar problemas
- **Commit:** `git commit -m "Lógica, funções e boas práticas - Dia 2"`

3.2 20h30 - 20h50 | INTERVALO

3.3 20h50 - 22h00 | Introdução ao Tidyverse

3.3.1 Filosofia Tidyverse (10min)

- Princípios: tidy data, pipe, verbos consistentes
- Carregar: `library(tidyverse)`

3.3.2 Verbos Essenciais do dplyr (45min)

- **`filter()`:** filtrar linhas por condição

- **`select()`:** selecionar colunas
- **`mutate()`:** criar/modificar colunas
- **`arrange()`:** ordenar linhas
- **`summarize()`:** resumir dados
- **`group_by()`:** agrupar para operações
- **Operador pipe:** `%>%` (tidyverse) ou `|>` (R nativo 4.1+)

3.3.3 Datas com lubridate (10min)

- Reconhecer datas: `ymd()`, `dmy()`, `mdy()`
- Extrair componentes: `year()`, `month()`, `wday()`
- Operar com datas: `today()`, `now()` e intervalos simples

3.3.4 Exemplo Integrado (20min)

- Dataset: dados de saúde pública ou agricultura
- Pipeline com todos os verbos; se houver coluna de data, usar `lubridate` para extrair mês/ano

3.3.5 Para Casa

- Praticar pipeline com dataset fornecido
- Documentar código com comentários

---

4 Dia 3: 24/11 (Segunda) - Manipulação Avançada e Visualização

4.1 19h00 - 20h30 | Transformação e I/O de Dados

4.1.1 Tidyr para Reshape (25min)

- **`pivot_longer()`:** wide → long
- **`pivot_wider()`:** long → wide
- **`separate()`** e **`unite()`**

4.1.2 Tratamento de Valores Ausentes (20min)

- Identificar: `is.na()`, `sum(is.na())`
- Remover: `drop_na()`, `na.omit()`
- Imputar: `replace_na()`, `fill()`

4.1.3 Leitura e Escrita de Arquivos (35min)

```r
# CSV com encoding correto (para acentuação)
# Windows (Latin1) vs macOS/Linux (UTF-8)
library(readr)

# Detectar encoding automaticamente
dados <- read_csv(here("data", "raw", "dados.csv"),
                  locale = locale(encoding = "UTF-8"))

# Para arquivos Windows com acentos problemáticos
dados <- read_csv(here("data", "raw", "dados.csv"),
                  locale = locale(encoding = "latin1"))

# Excel - funciona igual em todos os sistemas
library(readxl)
dados <- read_excel(here("data", "raw", "planilha.xlsx"),
                    sheet = 1)

# Salvar sempre em UTF-8 (padrão universal)
write_csv(dados, here("output", "resultado.csv"))
```

4.1.3.1 Encoding e Separadores (IMPORTANTE!)

```r
# Brasil usa vírgula, EUA/UK usam ponto
# readr detecta automaticamente, mas pode especificar:
dados <- read_csv2(arquivo)  # Para separador ";"  e decimal ","
dados <- read_csv(arquivo)   # Para separador "," e decimal "."
```

4.1.3.2 Separadores de Decimal (Região)

```r
#  EVITE barras específicas de sistema
"C:\\Users\\nome\\arquivo.csv"      # Windows (barra invertida)
"/Users/nome/arquivo.csv"          # macOS/Linux (barra normal)

#  USE here::here() (funciona em TODOS)
library(here)
arquivo <- here("data", "raw", "dados.csv")

# here() constrói o caminho correto para seu sistema!
# Windows: "C:\\Users\\...\\curso\\data\\raw\\dados.csv"
# macOS: "/Users/.../curso/data/raw/dados.csv"
# Linux: "/home/.../curso/data/raw/dados.csv"
```

4.1.3.3 Caminhos de Arquivos (Portabilidade)

4.1.4 Organização de Projetos (10min)

```
meu-projeto/
  meu-projeto.Rproj    # SEMPRE use .Rproj!
  README.md
  data/
     raw/             # Originais (nunca modificar)
     processed/       # Limpos
  scripts/
     01_import.R
     02_clean.R
     03_analyze.R
  output/
     figures/
     tables/
  docs/
```

Vantagens de usar Projetos RStudio: - Working directory automático - Portabilidade entre sistemas - Funciona perfeitamente com Git - `here::here()` sempre encontra a raiz do projeto

4.1.5 Ferramentas Modernas (10min)

```r
library(janitor)
dados <- clean_names(dados)  # Padroniza nomes de colunas

library(skimr)
skim(dados)  # Resumo estatístico completo
```

4.1.6 Pipeline integrado com palmerpenguins (25min)

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

```r
library(ggplot2)
library(scales)     # para formatação de eixos
library(patchwork)  # para combinar gráficos

peng <- penguins |>
  filter(!is.na(bill_length_mm), !is.na(bill_depth_mm), !is.na(flipper_length_mm)) |>
  mutate(raz_bico = bill_length_mm / bill_depth_mm)

g1 <- ggplot(peng, aes(x = species, y = flipper_length_mm)) +
  geom_boxplot() +
  labs(x = "Espécie", y = "Comprimento da nadadeira (mm)",
       title = "Distribuição por espécie")

g2 <- ggplot(peng, aes(x = bill_length_mm, y = bill_depth_mm, color = species)) +
  geom_point(alpha = 0.7) +
  labs(x = "Comprimento do bico (mm)", y = "Profundidade do bico (mm)",
       title = "Relação entre medidas do bico")

# Combinar lado a lado
g1 + g2
```

```r
library(here)
# Exemplo scales: eixos percentuais ou monetários (quando aplicável)
# scale_y_continuous(labels = percent)
# scale_y_continuous(labels = dollar())

ggsave(here("output", "figures", "penguins_combined.png"), width = 10, height = 4, dpi = 300)
```

4.1.6.1 Exportação com here() e formatação com scales (5min) **Commit:** `git commit -m "Transformação, I/O e pipeline - Dia 3"`

4.2 20h30 - 20h50 | INTERVALO

4.3 20h50 - 22h00 | Visualização com ggplot2

4.3.1 Gramática de Gráficos (15min)

- Filosofia: camadas sequenciais
- Estrutura: `ggplot(data, aes()) + geom_*()`
- **Aesthetics:** `x`, `y`, `color`, `fill`, `size`

4.3.2 Tipos de Gráficos (50min)

- **Dispersão:** `geom_point()` + `geom_smooth()`

- **Barras:** `geom_col()` vs. `geom_bar()`
- **Boxplot:** `geom_boxplot()` + `geom_jitter()`
- **Linhas:** `geom_line()` (séries temporais)
- **Histograma:** `geom_histogram()` + `geom_density()`
- **Formatação de eixos com scales:** `percent`, `number`, `dollar` (quando fizer sentido)
- **Combinação de gráficos com patchwork:** `g1 + g2`, `g1 / g2`

4.3.3 Personalização e Salvamento (15min)

```
# Criar gráfico
p <- ggplot(peng, aes(x = bill_length_mm, y = flipper_length_mm, color = species)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Relação entre bico e nadadeira",
       x = "Comprimento do bico (mm)",
       y = "Comprimento da nadadeira (mm)") +
  scale_y_continuous(labels = number)  # formatação amigável do eixo y

# Salvar com caminho portável
ggsave(here("output", "figures", "grafico.png"),
       plot = p,
       width = 8,
       height = 6,
       dpi = 300)
```

4.3.4 Prática (10min)

- Criar 3 visualizações diferentes
- Salvar usando `ggsave()` e `here()`
- **Commit:** `git commit -m "Visualização com ggplot2 - Dia 3"`

---

5 Dia 4: 25/11 (Terça) - RMarkdown e (Opcional) Integração de IA

5.1 19h00 - 20h30 | Relatórios Reprodutíveis com RMarkdown

5.1.1 Introdução à Programação Literária (15min)

- **Conceito:** código + narrativa + resultados
- **Reprodutibilidade entre sistemas**
- Criar documento: File → New File → R Markdown

5.1.2 Estrutura de Documentos (35min)

```
---
title: "Minha Análise"
author: "Seu Nome"
date: "2025-10-07"
output:
  html_document:
    toc: true
    toc_float: true
    code_folding: show
---
```

Markdown Básico (universal): - Headers: `#`, `##`, `###` - Listas: `-` ou `1.` - Ênfase: `*itálico*`, `**negrito**` - Links: `[texto](url)` - Imagens: `![descrição](caminho)`

Chunks de código:

```
```r
# Use here() para caminhos portáveis!
dados <- read_csv(here("data", "raw", "dados.csv"))
```
```

5.1.3 Renderização (20min)

Knit para diferentes formatos: - **HTML:** Funciona em todos os sistemas (padrão) - **PDF:** Requer LaTeX (pode ter problemas) - Windows: MiKTeX (https://miktex.org/) - macOS: MacTeX (https://www.tug.org/mactex/) - Linux: `sudo apt install texlive-full` - **Alternativa:** `install.packages("tinytex"); tinytex::install_tinytex()` - **Word:** Funciona em todos (requer MS Word instalado para visualizar)

Dica: Começe com HTML (sempre funciona), depois experimente PDF.

```
# Renderizar programaticamente (qualquer sistema)
rmarkdown::render(here("docs", "relatorio.Rmd"),
                  output_format = "html_document")
```

5.1.4 GitHub: Documentação Final (20min)

Comandos Git (multiplataforma via RStudio Terminal):

```
git add .
git commit -m "Relatório RMarkdown - Dia 4"
git push
```

Arquivo .gitignore (funciona em todos os sistemas):

```
# R files
.Rproj.user
.Rhistory
.RData
.Ruserdata

# Output files
*.html
*.pdf
output/

# System files (importantes!)
.DS_Store      # macOS
Thumbs.db      # Windows
*~             # Linux
```

README.md - Template Portável:

```
# Nome do Projeto

## Sobre
Este é meu repositório pessoal para o curso "Introdução à Programação em R com GitHub, ChatGPT

**Repositório original do curso:** https://github.com/viniciusjunqueira/curso-r-github-ia

## Como Reproduzir Esta Análise

### Pré-requisitos
- R 4.3+
- RStudio
- Pacotes: tidyverse, here, rmarkdown

### Instalação
```r
install.packages(c("tidyverse", "here", "rmarkdown"))
```

5.1.5 Execução

1. Clone este repositório (ou faça seu próprio fork)
2. Abra o arquivo .Rproj
3. Execute os scripts na ordem:
    - `scripts/01_import.R`
    - `scripts/02_clean.R`
```

- `scripts/03_analyze.R`

Nota: Funciona em Windows, macOS e Linux!

5.2 Estrutura do Projeto

```
meu-projeto/
   data/raw/       # Dados originais
   data/processed/ # Dados processados
   scripts/        # Scripts R
   output/         # Resultados e gráficos
   docs/           # Documentação
```

5.3 Contato

[Seu nome e informações de contato]

````
---

## 20h30 - 20h50 | INTERVALO

---

## 20h50 - 22h00 | (Opcional) Integração de IA no RStudio

### Visão Geral (10min)
- Por que integrar IA ao RStudio?
- Ferramentas disponíveis: gptstudio, chattr, APIs
- **Compatibilidade:** Todas funcionam em Windows, macOS e Linux

### gptstudio: ChatGPT no RStudio (30min)

#### Instalação (Multiplataforma)
```r
# Funciona em todos os sistemas
install.packages("gptstudio")
library(gptstudio)

# Configurar API Key (todos os sistemas)
# Opção 1: Temporária (essa sessão)
Sys.setenv(OPENAI_API_KEY = "sua-chave-aqui")

# Opção 2: Permanente (recomendado)
usethis::edit_r_environ()  # Abre arquivo para editar
# Adicionar linha: OPENAI_API_KEY=sua-chave
# Salvar e reiniciar R
````
````

5.3.0.1 Funcionalidades (20min)

Todas as funções aparecem em: **Addins** (menu superior do RStudio)

- **Comentar código:** Selecione código → Addins → "Comment your code"
- **Escrever código:** Escreva comentário → Addins → "Write code"
- **Explicar código:** Selecione trecho → Addins → "Explain code"
- **Melhorar código:** Selecione → Addins → "Suggest improvements"
- **Chat interativo:** Addins → "ChatGPT in Source"

Nota: Funciona identicamente em Windows, macOS e Linux!

5.3.1 chattr: Múltiplos LLMs (20min)

```
# Multiplataforma
install.packages("chattr")
library(chattr)

# Para ChatGPT (OpenAI)
Sys.setenv(OPENAI_API_KEY = "sua-chave")
chattr_use("gpt4")

# Para Claude (Anthropic)
Sys.setenv(ANTHROPIC_API_KEY = "sua-chave")
chattr_use("anthropic")

# Configuração permanente
usethis::edit_r_environ()
# Adicionar:
# OPENAI_API_KEY=sua-chave
# ANTHROPIC_API_KEY=sua-chave
```

5.3.1.1 Instalação e Configuração

```
# Abrir interface de chat
chattr_app()

# Perguntas diretas no console
chattr("Como fazer um boxplot colorido no ggplot2?")

# Pedir código
chattr("Criar função que calcula desvio padrão")

# Debugging
chattr("Por que este código não funciona? [colar código]")
```

5.3.1.2 Uso Básico

5.3.2 API Direta: Claude via httr2 (15min)

```
# Funciona em todos os sistemas
library(httr2)

ask_claude <- function(prompt,
                       api_key = Sys.getenv("ANTHROPIC_API_KEY")) {
  request("https://api.anthropic.com/v1/messages") |>
    req_headers(
      `x-api-key` = api_key,
      `anthropic-version` = "2023-06-01",
      `content-type` = "application/json"
    ) |>
    req_body_json(list(
      model = "claude-3-5-sonnet-20241022",
      max_tokens = 1024,
      messages = list(list(
        role = "user",
        content = prompt
      ))
    )) |>
    req_perform() |>
    resp_body_json()
}

# Usar (funciona em qualquer SO)
resposta <- ask_claude("Explique o que é tidy data")
cat(resposta$content[[1]]$text)
```

5.3.3 Boas Práticas com IA (10min)

5.3.3.1 Considerações Importantes

- **Segurança:** NUNCA envie dados sensíveis/confidenciais
- **Validação:** SEMPRE teste código gerado
- **Aprendizado:** Entenda antes de usar
- **Custos:** APIs pagas - monitore uso

5.3.3.2 Workflow Recomendado

1. Aprenda os conceitos básicos primeiro
2. Use IA para explicações e debugging
3. Valide e teste todo código gerado
4. Documente usando IA como assistente

5.3.4 Bônus rápido: resultados tidy com broom (10min)

```
library(broom)
mod <- lm(bill_length_mm ~ bill_depth_mm + flipper_length_mm, data = penguins)
```

```r
tidy(mod)       # tabela tidy de coeficientes
glance(mod)     # métricas de ajuste (R² etc.)
augment(mod)    # dados originais + ajustados/resíduos
```

5.3.5 Prática Final (25min)

1. Configurar gptstudio OU chattr
2. Usar IA para:
   - Comentar script anterior
   - Criar nova função
   - Otimizar pipeline
   - Melhorar relatório RMarkdown
3. Salvar tudo usando `here()`
4. **Commit final:** `git commit -m "IA integrada - Curso completo"`

5.3.6 Recursos (5min)

- **gptstudio:** https://github.com/MichelNivard/gptstudio
- **chattr:** https://mlverse.github.io/chattr/
- **OpenAI API:** https://platform.openai.com/
- **Anthropic API:** https://console.anthropic.com/

5.3.7 Encerramento (5min)

- Comunidades: R-Ladies, Posit Community
- Prática contínua
- Feedback do curso

---

6 Troubleshooting por Sistema Operacional

6.1 Windows

6.1.1 Problema: Git não encontrado

Solução: - Reinstalar Git: https://git-scm.com/download/win - Marcar opção "Git from the command line and also from 3rd-party software" - Reiniciar RStudio

6.1.2 Problema: Acentos estranhos nos dados

Solução:

```
# Usar encoding correto
dados <- read_csv(arquivo, locale = locale(encoding = "latin1"))
```

6.1.3 Problema: Erro ao instalar pacotes

Solução: - Executar RStudio como Administrador - Ou instalar em biblioteca de usuário (padrão)

6.2 macOS

6.2.1 Problema: "xcrun: error" ao usar Git

Solução:

```
xcode-select --install
```

6.2.2 Problema: LaTeX não encontrado para PDF

Solução:

```
install.packages("tinytex")
tinytex::install_tinytex()
```

6.2.3 Problema: Permissões negadas

Solução: - Verificar configurações de segurança - System Preferences → Security & Privacy

6.3 Linux (Ubuntu/Debian)

6.3.1 Problema: Erro ao instalar pacotes R

Solução:

```
# Instalar dependências de desenvolvimento
sudo apt install build-essential libcurl4-openssl-dev
sudo apt install libssl-dev libxml2-dev
sudo apt install libfontconfig1-dev libharfbuzz-dev libfribidi-dev
```

6.3.2 Problema: Git não configurado

Solução:

```
sudo apt update
sudo apt install git
git config --global user.name "Seu Nome"
git config --global user.email "email@exemplo.com"
```

6.4 Problemas Comuns (Todos os Sistemas)

6.4.1 "Permission denied" ao fazer push

Causa: Você clonou o repositório original ao invés do seu fork

Como verificar:

```
git remote -v
# Se aparecer "viniciusjunqueira", você clonou o errado!
```

Solução - Opção 1 (Reconfigurar remote):

```
# Apontar para SEU fork
git remote set-url origin https://github.com/SEU-USUARIO/curso-r-github-ia.git

# Verificar
git remote -v
# Agora deve aparecer SEU-USUARIO

# Tentar push novamente
git push origin main
```

Solução - Opção 2 (Recomeçar do zero): 1. Deletar pasta do projeto
2. Fazer fork do repositório original (se ainda não fez)
3. Clonar SEU fork corretamente

6.4.2 "Não consigo encontrar o botão Fork"

Solução: - Certifique-se que está logado no GitHub
- Acesse: https://github.com/viniciusjunqueira/curso-r-github-ia
- O botão "Fork" fica no canto superior direito, ao lado de "Star"

6.4.3 "Already exists" ao criar fork

Causa: Você já tem um fork deste repositório

Solução: - Vá direto para: `https://github.com/SEU-USUARIO/curso-r-github-ia`
- Clone este repositório (você já tem o fork!)

6.4.4 "Não sei meu nome de usuário do GitHub"

Solução: 1. Acesse: https://github.com
2. Clique no seu avatar (canto superior direito)
3. Seu nome de usuário aparece no topo do menu
4. Ou veja na URL: `https://github.com/SEU-USUARIO`

6.4.5 "Pacote não disponível"

```r
# Tentar espelho diferente
options(repos = "https://cloud.r-project.org/")
install.packages("nome_do_pacote")
```

6.4.6 "API Key não encontrada"

```r
# Verificar se está configurada
Sys.getenv("OPENAI_API_KEY")

# Se vazio, configurar:
usethis::edit_r_environ()
# Adicionar: OPENAI_API_KEY=sua-chave
# Salvar e reiniciar R: .rs.restartR()
```

6.4.7 "Erro de encoding"

```r
# Sempre usar UTF-8
write_csv(dados, here("output", "dados.csv"))
# readr usa UTF-8 por padrão
```

---

7 Recursos Adicionais

7.1 Materiais Multiplataforma

7.1.1 Livros Online (Todos os Sistemas)

- **R for Data Science (2e):** https://r4ds.hadley.nz/
- **Happy Git with R:** https://happygitwithr.com/
- **R Graphics Cookbook:** https://r-graphics.org/

7.1.2 Comunidades

- **Posit Community:** https://community.rstudio.com/
- **Stack Overflow:** tag [r]
- **R-Ladies:** https://rladies.org/

7.1.3 Datasets para Prática

- **tidytuesday:** https://github.com/rfordatascience/tidytuesday
- **Brasil.io:** https://brasil.io/datasets/
- **Kaggle:** https://www.kaggle.com/datasets

7.2 Obtendo API Keys (Todos os Sistemas)

7.2.1 OpenAI (ChatGPT)

1. https://platform.openai.com/signup
2. API Keys → Create new secret key
3. Copiar e guardar a chave
4. Configurar no R:

```
usethis::edit_r_environ()
# Adicionar: OPENAI_API_KEY=sk-...
```

7.2.2 Anthropic (Claude)

1. https://console.anthropic.com/
2. Settings → API Keys → Create Key
3. Configurar:

```
usethis::edit_r_environ()
# Adicionar: ANTHROPIC_API_KEY=sk-ant-...
```

7.2.3 Alternativas Gratuitas

- **Ollama:** Modelos locais (sem internet!)
  - Windows: https://ollama.ai/download/windows
  - macOS: https://ollama.ai/download/mac
  - Linux: https://ollama.ai/download/linux
  - Funciona com chattr

7.3 Contato

Instrutor: Vinícius Silva Junqueira
Email: junqueiravinicius@hotmail.com
GitHub: https://github.com/viniciusjunqueira/curso-r-github-ia
Lattes: http://lattes.cnpq.br/4686677580216927
LinkedIn: www.linkedin.com/in/junqueiravinicius

Última atualização: 2025-10-07
Versão: 5.1 - Núcleo ajustado + IA opcional + pipeline integrado com palmerpenguins