Aula 05 – Visualização de Dados com ggplot2 - Curso R para iniciantes

Adhemar Ranciaro Neto

Aula 05 – Visualização de Dados com ggplot2

Nesta aula, aprenderemos a criar gráficos de barras, linhas, setores (pizza) e histogramas usando o pacote `ggplot2`.

Carregando pacotes

```
library(tidyverse)   # dplyr, ggplot2, readr, etc.

# Bases que já vêm com ggplot2 e datasets:
# - mpg (consumo de carros nos EUA)
# - diamonds (preços e características de diamantes)
# - economics (série temporal macroeconômica)
# - mtcars (carros - vem no pacote 'datasets' base do R)
```

Espiando dados

Usaremos bases de dados incluídas em pacotes do R

```
glimpse(mpg)
glimpse(diamonds)
glimpse(economics)
glimpse(mtcars)

view(mtcars)
view(diamonds)
view(mpg)
```

Gráfico de barras

Exemplo: quantos carros por classe (class) na base mpg.

```
mpg %>%
  count(class, name = "n") %>%
  mutate(class = fct_reorder(class, n)) %>%  #ordena pelas contagens
  ggplot(aes(x = class, y = n)) +
  geom_col(fill = "green") +
  geom_text(aes(label = n), vjust = -0.4) +
```

```
  labs(title = "Quantidade de carros por classe",
       x = "Classe",
       y = "Contagem") +
  theme_minimal()
```

Gráfico de barras em ordem decrescente

```
mpg %>%
  count(class, name = "n") %>%
  mutate(class = fct_reorder(class, -n)) %>%  #ordena pelas contagens
  ggplot(aes(x = class, y = n)) +
  geom_col(fill = "green") +
  geom_text(aes(label = n), vjust = -0.4) +
  labs(title = "Quantidade de carros por classe",
       x = "Classe",
       y = "Contagem") +
  theme_minimal()
```

Boas práticas:

use `count()` em vez de `group_by() + summarise(n = n())`;

use `fct_reorder()` para ordenar as barras;

rótulos com `geom_text()` facilitam a leitura.

Barras proporcionais (percentuais)

```
mpg %>%
  count(class, name = "n") %>%
  mutate(pct = n / sum(n)) %>%
  ggplot(aes(x = fct_reorder(class, pct), y = pct)) +
  geom_col(fill = "tomato") +
  scale_y_continuous(labels = scales::percent) +
  geom_text(aes(label = round(pct,3)*100), vjust = -0.4) +
  labs(title = "Proporção por classe de veículo",
       x = "Classe",
       y = "Proporção") +
  theme_classic()
```

Gráfico de linhas

A base economics tem colunas date e unemploy (nº de desempregados). Vamos ver a evolução:

```
economics %>%
  ggplot(aes(x = date, y = unemploy)) +
  geom_line(size = 1) +
  labs(title = "Evolução do desemprego nos EUA",
       x = "Ano",
       y = "Número de desempregados") +
  scale_y_continuous(labels = scales::label_comma()) +
  theme_bw()
```

```r
economics %>%
  ggplot(aes(x = date, y = unemploy)) +
  geom_line(color = "blue", size = 1, linetype = "dotted") +
  labs(title = "Evolução do desemprego nos EUA",
       x = "Ano",
       y = "Número de desempregados") +
  scale_y_continuous(labels = scales::label_comma()) +
  theme_bw()
```

Linha com suavização (tendência)

```r
economics %>%
  ggplot(aes(x = date, y = unemploy)) +
  geom_line(alpha = 0.5) +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Desemprego com linha de tendência (LOESS)",
       x = "Ano", y = "Desemprego") +
  theme_minimal()
```

Métodos geom_smooth

"auto" (padrão): escolhe automaticamente loess para amostras menores e gam para maiores.

"loess": suavização local (boa para curvas flexíveis com poucos pontos; fica lenta/instável com muitos dados).

"lm": regressão linear (reta ou polinômio se você ajustar a fórmula).

"glm": regressão linear generalizada (precisa informar a família, ex.: binomial).

"gam": regressão aditiva generalizada (suavização via splines, requer mgcv).

"rlm": regressão linear robusta (resistente a outliers, requer MASS).

Gráfico de setores (pizza)

Observação didática: pizza não é a melhor forma para comparar categorias; barras proporcionais ou barras empilhadas costumam ser mais legíveis. Mas é útil conhecer.

```r
mpg %>%
  count(class, name = "n") %>%
  mutate(pct = n / sum(n)) %>%
  ggplot(aes(x = "", y = pct, fill = class)) +
  geom_col(width = 1, color = "white") +
  coord_polar(theta = "y") +
  geom_text(aes(label = scales::percent(pct)),
            position = position_stack(vjust = 0.5), size = 4) +
  labs(title = "Distribuição por classe (mpg)", x = NULL, y = NULL, fill = "Classe") +
  theme_void()
```

Barras horizontais

```r
mpg %>%
  count(class, name = "n") %>%
  mutate(pct = n / sum(n),
         class = fct_reorder(class, pct)) %>%
  ggplot(aes(x = class, y = pct)) +
  geom_col() +
```

```
  coord_flip() +
  scale_y_continuous(labels = scales::percent) +
  labs(title = "Proporção por classe (alternativa às pizzas)",
       x = "Classe", y = "Proporção") +
  theme_minimal()
```

Histograma

Vamos usar diamonds$price (preço do diamante). Histograma mostra a distribuição de um numérico.

```
ggplot(diamonds, aes(x = price)) +
  geom_histogram(binwidth = 500, color = "white", fill = "steelblue") +
  labs(title = "Distribuição dos preços de diamantes",
       x = "Preço (USD)", y = "Frequência") +
  theme_minimal()
```

Densidade (alternativa suave)

```
ggplot(diamonds, aes(x = price)) +
  geom_density(fill = "lightblue", alpha = 0.6) +
  labs(title = "Densidade dos preços de diamantes",
       x = "Preço (USD)", y = "Densidade") +
  theme_classic()
```

Facetas (comparar grupos lado a lado)

Quantos carros por fabricante (manufacturer), separados por tipo de combustível (fl) na base mpg:

```
mpg %>%
  count(manufacturer, fl) %>%
  mutate(manufacturer = fct_reorder(manufacturer, n, .fun = sum)) %>%
  ggplot(aes(x = manufacturer, y = n)) +
  geom_col() +
  facet_wrap(~ fl, scales = "free_y") +
  coord_flip() +
  labs(title = "Contagem por fabricante, facetando por combustível",
       x = "Fabricante", y = "Contagem") +
  theme_bw()
```

Gráfico de dispersão

```
mtcars %>%
  rownames_to_column("modelo") %>%
  ggplot(aes(x = wt, y = mpg, color='red')) +
  geom_point(size = 3) +
  labs(title = "Consumo (mpg) vs Peso (wt)",
       x = "Peso (1000 lbs)", y = "Milhas por galão")+
  theme_minimal()
```

Mapear estética por grupo (cores, formas)

Relação entre peso e consumo na base mtcars:

```r
mtcars %>%
  rownames_to_column("modelo") %>%
  mutate(cilindros = factor(cyl)) %>%
  ggplot(aes(x = wt, y = mpg, color = cilindros)) +
  geom_point(size = 3) +
  geom_smooth(se = FALSE) +
  labs(title = "Consumo (mpg) vs Peso (wt) por nº de cilindros",
       x = "Peso (1000 lbs)", y = "Milhas por galão",
       color = "Cilindros") +
  theme_minimal()
```

Títulos, rótulos e escalas

Exemplo rápido em mpg, melhorando rótulos:

```r
mpg %>%
  mutate(cambio = if_else(trans %>% str_detect("^auto"), "Automático", "Manual")) %>%
  count(cambio) %>%
  ggplot(aes(x = cambio, y = n, fill = cambio)) +
  geom_col(show.legend = FALSE) +
  geom_text(aes(label = n), vjust = -0.4) +
  labs(title = "Transmissão dos veículos (mpg)",
       x = NULL, y = "Quantidade") +
  scale_y_continuous(expand = expansion(mult = c(0, .1))) +
  theme_minimal()
```

Exportando gráficos

```r
p <- ggplot(diamonds, aes(x = price)) +
  geom_histogram(binwidth = 500, color = "white", fill = "steelblue") +
  labs(title = "Distribuição dos preços de diamantes",
       x = "Preço (USD)", y = "Frequência") +
  theme_minimal()

ggsave("hist_preco_diamonds.png", p, width = 8, height = 5, dpi = 300)
```

Mini Desafios

1. Barras (mpg): faça um gráfico com a soma de carros por manufacturer. Ordene as barras e coloque rótulos.
2. Pizza (mpg): mostre a proporção de drv (tração: f, r, 4). Em seguida, refaça como barras proporcionais (qual ficou mais legível?).

```r
mpg %>%
  count(drv, name = "n") %>%
  mutate(pct = n / sum(n)) %>%
  ggplot(aes(x = "", y = pct, fill = drv)) +
```

```
  geom_col(width = 1, color = "white") +
  coord_polar(theta = "y") +
  geom_text(aes(label = scales::percent(pct)),
            position = position_stack(vjust = 0.5), size = 4) +
  labs(title = "Distribuição por tipo de tração (mpg)", x = NULL, y = NULL, fill = "Tração") +
  theme_void()
```

Gráfico de barras horizontais

```
mpg %>%
  count(drv, name = "n") %>%
  mutate(pct = n / sum(n),
         class = fct_reorder(drv, pct)) %>%
  ggplot(aes(x = drv, y = pct)) +
  geom_col() +
  coord_flip() +
  scale_y_continuous(labels = scales::percent) +
  geom_text(aes(label = round(pct,3)*100), vjust = +0.4, hjust = -0.5) +
  labs(title = "Proporção por tração (alternativa às pizzas)",
       x = "Tração", y = "Proporção") +
  theme_minimal()
```

3. Histograma (diamonds): plote a distribuição de carat. Teste dois valores de binwidth e comente o impacto.
4. Linhas (economics): plote psavert (taxa de poupança pessoal) ao longo do tempo. Adicione uma linha de tendência.

Tarefa para casa

1. Comparação por grupo (`mpg`): crie um gráfico comparando a contagem por `class` separando por `cyl` (facetas ou cores).
2. Exportação: salve um gráfico seu como "meu_grafico.png" com ggsave().
3. Desafio bônus: no `diamonds`, crie um boxplot de `price` por `cut` com `coord_flip()` e rótulos de milhares (`scales::label_comma()`).

Resumo da Aula

- Aprendemos a usar `ggplot2` para criar gráficos:
  - Barras: `geom_col()` (e `count()` para sumarizar)
  - Linhas: `geom_line()` para séries temporais (ex.: `economics`)
  - Pizza: `coord_polar(theta = "y")` (use com moderação; barras proporcionais são mais legíveis)
  - Histograma: `geom_histogram()` (ajuste binwidth)

Boas práticas: ordenar fatores (`fct_reorder`), formatar eixos (`scales`), usar temas (`theme_*`), facetar (`facet_*`) e exportar (`ggsave()`)

COMPLEMENTO

1. Melhorando estética com temas e cores

O ggplot2 permite personalizar aparência, usar paletas e temas prontos:

```
library(tidyverse)

# Exemplo: preço x quilates em diamonds, colorindo por corte
ggplot(diamonds, aes(x = carat, y = price, color = cut)) +
  geom_point(alpha = 0.6) +
  labs(title = "Preço vs Peso de diamantes",
       x = "Peso (carat)", y = "Preço (USD)") +
  theme_minimal(base_size = 12) +
  theme(legend.position = "bottom")
```

Temas embutidos:

```
+ theme_classic()
+ theme_light()
+ theme_bw()
+ theme_dark()
+ theme_void()
```

Cores personalizadas ou escalas:

```
+ scale_color_brewer(palette = "Set2")
# ou
+ scale_color_viridis_d()
```

---

2. Facetas (comparações lado a lado)

Permite comparar grupos facilmente — ótimo para explicar painéis múltiplos.

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_wrap(~ class) +
  labs(title = "Relação entre cilindrada e consumo por classe") +
  theme_minimal()
```

`facet_wrap(~variável)` cria vários gráficos com a mesma escala.

---

3. Boxplots e violinos (comparando distribuições)

Excelente para comparar distribuições numéricas por grupos:

```
ggplot(diamonds, aes(x = cut, y = price)) +
  geom_boxplot(fill = "lightblue") +
  labs(title = "Distribuição de preço por tipo de corte",
       x = "Tipo de corte", y = "Preço (USD)") +
  coord_flip()
```

Complemento visual moderno (gráfico de violino):

```
+ geom_violin(fill = "skyblue", alpha = 0.4)
```

```
ggplot(diamonds, aes(x = cut, y = price)) +
    geom_violin(fill = "skyblue", alpha = 0.4) +
    labs(title = "Distribuição de preço por tipo de corte",
       x = "Tipo de corte", y = "Preço (USD)") +
  coord_flip()
```

---

4. Gráficos de dispersão com tendência

Importância de observar relações entre variáveis:

```
ggplot(mpg, aes(x = displ, y = hwy, color = class)) +
  geom_point(size = 3, alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Cilindrada vs Consumo por classe",
       x = "Cilindrada", y = "Consumo (mpg)") +
  theme_minimal()
```

Conceito: `geom_point()` mostra dados individuais; `geom_smooth()` ajuda a visualizar a tendência média.

---

5. Dicas práticas para legibilidade e layout

Boas práticas:

Evitar sobreposição de texto (`vjust`, `hjust`, `position_dodge()`).

Reduzir ruído visual (`theme_minimal()` ou `theme_light()`).

Usar rótulos e legendas claras (`labs()`).

Limitar eixos: `xlim()`, `ylim()`.

Exemplos com rótulos deslocados:

```
mpg %>%
  count(class, name = "n") %>%
  mutate(class = fct_reorder(class, n)) %>%
  ggplot(aes(x = class, y = n)) +
  geom_col(fill = "steelblue") +
  geom_text(aes(label = n), vjust = -0.25, size = 4) +
  scale_y_continuous(expand = expansion(mult = c(0, .10))) +  # folga no topo
  labs(title = "Número de carros por classe",
       x = "Classe", y = "Contagem") +
  theme_minimal()
```

```
mpg %>%
  count(class) %>%
  mutate(class = fct_reorder(class, n)) %>%
  ggplot(aes(x = class, y = n)) +
  geom_col(fill = "steelblue") +
  geom_text(aes(label = n), vjust = -0.2, size = 4) +
  scale_y_continuous(expand = expansion(mult = c(0, .1))) +
  labs(title = "Número de carros por classe",
```

```
       x = "Classe", y = "Contagem") +
  theme_minimal()
```

---

6. Introdução a escalas logarítmicas e transformações

Podemos mudar as escalas dos eixos:

```
ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(alpha = 0.5) +
  scale_y_log10(labels = scales::comma) +
  labs(title = "Preço vs Peso (escala logarítmica)",
       x = "Peso (carat)", y = "Preço (log10)") +
  theme_light()
```

Ajuda a ver relações "não lineares" e reduzir assimetria.

---

7. Gráficos combinados e camadas múltiplas

Veja o poder do ggplot2 em sobrepor diferentes tipos de geom:

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(aes(color = class), size = 3, alpha = 0.7) +
  geom_smooth(method = "loess", se = FALSE, color = "black", linetype = "dashed") +
  labs(title = "Dispersão com linha de suavização",
       x = "Cilindrada", y = "Consumo (mpg)") +
  theme_minimal()
```

---

8. Exportação profissional de gráficos

Salvando gráficos com qualidade:

```
p <- ggplot(diamonds, aes(x = price)) +
  geom_histogram(binwidth = 500, fill = "tomato", color = "white") +
  labs(title = "Distribuição dos preços de diamantes")

ggsave("grafico_diamantes.png", plot = p,
       width = 8, height = 5, dpi = 300)
```

Faça o teste da diferença entre formatos (.png, .pdf, .svg) e resoluções (dpi). Observe o que fica melhor.

---

9. Desafios práticos (para consolidar)

Boxplot + pontos: use `mpg` e combine `geom_boxplot()` + `geom_jitter()`.

Facetas: crie gráficos de `hwy ~ displ` separados por `drv`.

Tema e cores: refaça um gráfico com `theme_dark()` e `viridis`.

Anotação: adicione texto no gráfico com `annotate("text", ...)`.

---

10. Vamos além: animação e interatividade

gganimate: cria animações com base em tempo ou grupos.

plotly: transforma ggplots em gráficos interativos (ggplotly()).

Exemplos básicos com plotly:

```
library(plotly)
p <- ggplot(mpg, aes(displ, hwy, color = class)) + geom_point()
ggplotly(p)
```

Passe o mouse nos pontos do gráfico. DÊ 1 ou 2 cliques em um elemento qualquer da legenda e veja o que acontece.

```
library(tidyverse)
library(plotly)

# Série temporal: desemprego (economics)
p <- ggplot(economics, aes(x = date, y = unemploy)) +
  geom_line() +
  labs(title = "Desemprego nos EUA", x = NULL, y = "Pessoas (mil)")

ggplotly(p) %>%
  layout(
    xaxis = list(
      rangeslider = list(visible = TRUE),    # barra de rolamento
      rangeselector = list(                   # botões rápidos (opcional)
        buttons = list(
          list(count = 6,  label = "6m", step = "month", stepmode = "backward"),
          list(count = 1,  label = "1a", step = "year",  stepmode = "backward"),
          list(count = 5,  label = "5a", step = "year",  stepmode = "backward"),
          list(step = "all", label = "Tudo")
        )
      )
    )
  )
```

O exemplo acima coloca barra de rolamento de tempo para série temporal.

Exemplos com gganimate

```
install.packages(c("gganimate", "gifski"))
```

Linha animada: desemprego ao longo do tempo

- Dados: ggplot2::economics (colunas date, unemploy, etc.)
- Técnica: transition_reveal(date) “desenha” a linha conforme o tempo avança.

```
library(tidyverse)
library(gganimate)

p1 <- economics %>%
  ggplot(aes(x = date, y = unemploy)) +
```

```
  geom_line(size = 1) +
  geom_point(size = 2) +
  labs(title = "Evolução do desemprego nos EUA",
       subtitle = "Período: {format(frame_along, '%Y-%m')}",
       x = "Ano", y = "Número de desempregados") +
  theme_minimal(base_size = 12) +
  transition_reveal(date)

# Visualizar no RStudio
animate(p1, nframes = 150, fps = 20, width = 800, height = 450, renderer = gifski_renderer())

# Salvar em GIF
anim_save("desemprego_animado.gif", animation = last_animation())
```

Dispersão animada: consumo pessoal (pce) vs desemprego

- Mostra a trajetória dos pontos ao longo do tempo.
- Técnica: transition_time(date) atualiza o frame para cada instante temporal; shadow_mark() deixa um rastro discreto.

```
library(tidyverse)
library(gganimate)

p_time <- economics %>%
  ggplot(aes(x = pce, y = unemploy)) +
  geom_point(size = 3, alpha = 0.8) +
  shadow_mark(alpha = 0.2, size = 2) +
  labs(title = "Relação PCE x Desemprego ao longo do tempo",
       subtitle = "Mês: {format(frame_time, '%Y-%m')}",
       x = "Consumo pessoal (PCE)", y = "Desemprego") +
  theme_light(base_size = 12) +
  transition_time(date) +
  ease_aes("linear")

animate(p_time, nframes = 160, fps = 20, width = 800, height = 450, renderer = gifski_renderer())
anim_save("pce_vs_unemploy_time.gif", animation = last_animation())
```

Notas

- transition_reveal(x): perfeito para linhas; revela progressivamente seguindo x.
- transition_time(t): atualiza o gráfico para o valor de t (tempo); bom para pontos, barras, etc.
- shadow_mark() / shadow_wake(): adicionam "rastro" para dar contexto do passado.
- Em R Markdown/Quarto, funciona bem; se o GIF não aparecer no HTML, garanta que o chunk não tenha fig.show='hide' e que você está usando o renderer gifski.