Programação para Ciência de Dados

Análise Exploratória de Dados (EDA) - Parte 2

Arthur Casals

30 de Outubro de 2025

Avisos

- Segunda entrega quinzenal: **AMANHÃ (31/10)**
- Interação via Zoom

Agenda

- Revisão Aula 07
- Correlação e Relações entre Variáveis
- Seaborn: Visualizações Avançadas
- Análise Multivariada
- Comunicando Insights

Revisão Aula 07: O que Aprendemos

Fundamentos de EDA:

1. Conhecendo os Dados

- head(), shape, info(), describe()
- Identificar valores faltantes
- Tipos de variáveis (numéricas vs categóricas)

2. Estatística Descritiva

- Tendência central: média, mediana, moda
- Dispersão: desvio padrão, IQR
- Quartis: Q1, Q2, Q3
- Outliers: regra $1.5 \times IQR$

Revisão: Gráficos Básicos

3. Visualização com Matplotlib:

Histograma	**Boxplot**	**Bar Chart**
Distribuição	Quartis + outliers	Categorias
Forma, centro, dispersão	Comparar grupos	Frequências

Hoje vamos além:

- Ver **relações** entre variáveis
- Usar **Seaborn** para gráficos mais bonitos
- **Comunicar** insights de forma efetiva

Estatísticas: Quando Usar Cada Uma

Guia rápido de decisão:

Situação	Use	Evite
Dados simétricos	Média + DP	—
Dados com outliers	Mediana + IQR	Média
Dados categóricos	Moda	Média/Mediana
Comparar grupos	Boxplot	Só números
Ver distribuição	Histograma	Só média

Regra de ouro:

Sempre visualize + calcule estatísticas

Um complementa o outro!

Objetivos da Aula de Hoje

Ao final desta aula, você será capaz de:

1. **Entender correlação:**
   - O que significa correlação positiva/negativa
   - Calcular e interpretar matriz de correlação
   - Saber que correlação $\neq$ causalidade
2. **Criar visualizações avançadas:**
   - Scatter plots para ver relações
   - Usar Seaborn para gráficos bonitos
   - Heatmaps, pairplots, violinplots
3. **Análise completa:**
   - Ver múltiplas variáveis juntas
   - Identificar padrões e insights
   - Comunicar descobertas efetivamente

Bloco 1

Relações entre Variáveis

Da Análise Individual para Relações

Aula passada: Análise univariada

- Uma variável por vez
- ”Como é a distribuição de X?”
- ”Qual a média de Y?”

Hoje: Análise bivariada

- Duas variáveis juntas
- ”X e Y estão relacionados?”
- ”Quando X aumenta, Y aumenta também?”

Por que isso importa?

- Descobrir padrões ocultos
- Prever uma variável usando outra
- Entender causas e efeitos
- Base para Machine Learning

Correlação: O que é?

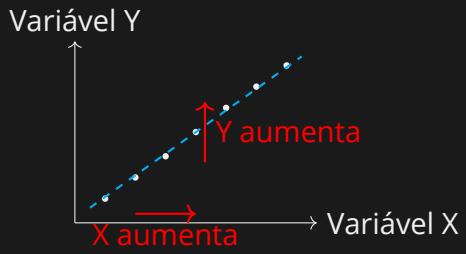
Correlação mede se duas coisas "andam juntas"

Exemplos do dia a dia:

- **Altura e peso:** Pessoas mais altas tendem a pesar mais
  - Correlação positiva ✓
- **Horas de estudo e notas:** Mais estudo $\rightarrow$ melhores notas
  - Correlação positiva ✓
- **Preço e demanda:** Preço sobe $\rightarrow$ demanda cai
  - Correlação negativa ✓
- **Cor do cabelo e nota:** Nenhuma relação
  - Sem correlação ✓

Correlação Positiva (Visual)

Quando uma variável sobe, a outra também sobe

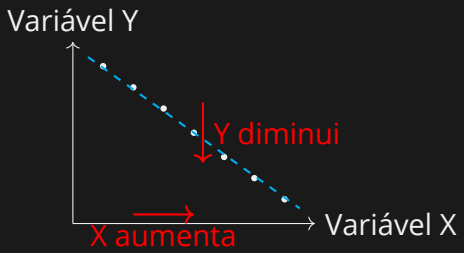


Exemplos:

- Temperatura vs vendas de sorvete
- Anos de experiência vs salário
- Tamanho da casa vs preço

Correlação Negativa (Visual)

Quando uma variável sobe, a outra desce

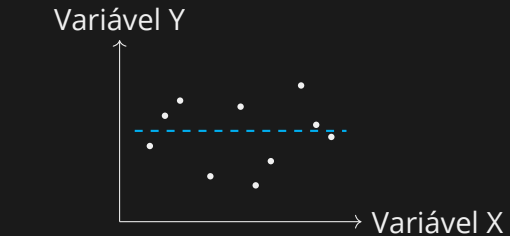


Exemplos:

- Preço vs quantidade vendida
- Distância vs frequência de visitas
- Idade do carro vs valor

Sem Correlação (Visual)

Nenhum padrão claro entre as variáveis



Pontos espalhados aleatoriamente

Exemplos:

- Tamanho do pé *vs* QI
- Signo do zodíaco *vs* desempenho profissional
- Cor favorita *vs* altura

Coeficiente de Pearson

O que o coeficiente r mede:

- Indica o quanto duas variáveis **variam juntas** de forma linear.
- Se r é alto, significa que quando uma muda, a outra tende a mudar de modo previsível.

Como entender intuitivamente:

- Se os pontos de um gráfico de dispersão estiverem próximos de uma linha reta $\rightarrow |r|$ é alto.
- Se estiverem espalhados $\rightarrow |r|$ é baixo.
- Sinal positivo: ambas aumentam juntas.
- Sinal negativo: uma aumenta, a outra diminui.

Em resumo: o coeficiente de Pearson diz *"quão reta" e "em que direção"* é a relação entre as variáveis.

Força da Correlação

Nem toda correlação é igualmente forte

Fraca

Padrão pouco claro
Muita dispersão
Difícil prever Y por X

Moderada

Padrão visível
Alguma dispersão
Previsão razoável

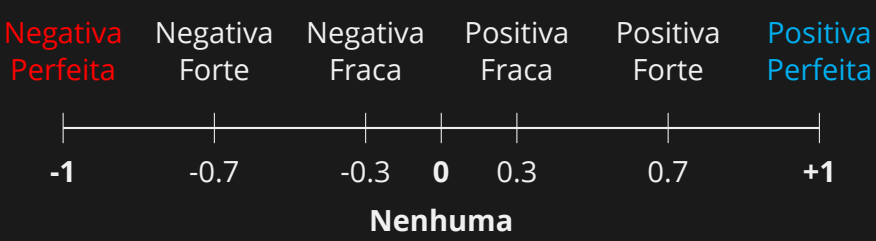
Forte

Padrão muito claro
Pouca dispersão
Boa previsão de Y por X

Medida numérica:

- Correlação varia de **-1 a +1**
- +1 = correlação positiva perfeita
- -1 = correlação negativa perfeita
- 0 = nenhuma correlação

Escala de Correlação: -1 a +1



Interpretação prática:

- $|r| > 0.7$: Forte (usável para previsões)
- $0.3 < |r| < 0.7$: Moderada (relação existe)
- $|r| < 0.3$: Fraca (relação pouco útil)

Coeficiente de Correlação de Pearson (r)

O que é:

- Mede a **intensidade** e a **direção** da relação linear entre duas variáveis quantitativas.
- Representado pela letra r, varia de -1 a $+1$.

Interpretação:

- $r > 0$: Relação positiva (ambas aumentam).
- $r < 0$: Relação negativa (uma aumenta, a outra diminui).
- $r = 0$: Sem relação linear.

Importante: Correlação *não implica causalidade!*

Fórmula de Pearson

A fórmula geral é:

$$r = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2 \sum (Y_i - \bar{Y})^2}}$$

Interpretação:

- O numerador mede a **covariância** entre X e Y.
- O denominador normaliza pelo desvio padrão das variáveis.
- Assim, r é uma versão *padronizada* da covariância.

Resultado: sempre $-1 \leq r \leq +1$.

Exemplos de Valores de *r*

r	**Força**	**Exemplo Prático**
+0.90	Muito forte	Altura e peso
+0.60	Moderada	Horas de estudo e nota
+0.25	Fraca	Exercício físico e qualidade do sono
0.00	Nenhuma	Signo e desempenho escolar
-0.70	Forte negativa	Preço e vendas

Dica visual:

- Quanto mais próximos os pontos estão da linha de tendência, mais forte é a correlação.

Correlação ≠ Causalidade!

AVISO CRÍTICO: Correlação NÃO implica causalidade

O que isso significa?

- X e Y podem estar correlacionados
- MAS X não necessariamente causa Y
- Pode haver uma terceira variável Z causando ambos
- Ou pode ser pura coincidência

⚠ Atenção

Correlação: ”X e Y andam juntos”
Causalidade: ”X causa Y”
São coisas MUITO diferentes!

Exemplo Clássico: Sorvete e Afogamentos

Observação:

Vendas de sorvete **correlacionam** com mortes por afogamento
(ambos aumentam no verão!)

Conclusão ERRADA:

- ✗ "Sorvete causa afogamentos"
- ✗ "Devemos banir sorvetes para salvar vidas"

Realidade:

- ✓ Temperatura alta causa:
  - Mais vendas de sorvete
  - Mais pessoas na praia/piscina
  - Mais afogamentos
- ✓ Variável oculta: **temperatura**

Exemplo Clássico: Sorvete e Afogamentos

⚠ Atenção

Sempre pergunte: ”Há uma terceira variável explicando ambos?”

Mais Exemplos de Correlação Espúria

Correlações reais, mas sem causalidade:

1. **Filmes com Nicolas Cage vs Afogamentos em piscinas**
   - Correlação: 0.67 (forte!)
   - Causalidade: Nenhuma (coincidência)
2. **Consumo de queijo vs Mortes em lençóis**
   - Correlação: 0.95 (muito forte!)
   - Causalidade: Zero (absurdo)
3. **Notas de alunos vs Tamanho do sapato**
   - Correlação positiva em dados reais!
   - Variável oculta: **idade** (crianças mais velhas = pés maiores + melhores notas)

Nota Importante

Site divertido: `https://tylervigen.com/spurious-correlations`

Calcular Correlação no Pandas

</> Python

```python
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
# Carregar Iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = iris.target_names[iris.target]
# Correlacao entre duas variaveis
corr = df['sepal length (cm)'].corr(df['petal length (cm)'])
print(f"Correlacao: {corr:.3f}")
# Output: Correlacao: 0.872

# Interpretação: Forte correlação positiva!
# Sepala maior tende a ter petala maior
```

Matriz de Correlação

Ver correlação entre TODAS as variáveis de uma vez

</> Python

```python
# Selecionar apenas colunas numericas
numeric_cols = ['sepal length (cm)', 'sepal width (cm)',
                'petal length (cm)', 'petal width (cm)']
# Calcular matriz de correlacao
corr_matrix = df[numeric_cols].corr()
print("=== MATRIZ DE CORRELACAO ===")
print(corr_matrix.round(2))
#                   sepal_len  sepal_wid  petal_len  petal_wid
# sepal length         1.00      -0.12       0.87       0.82
# sepal width         -0.12       1.00      -0.43      -0.37
# petal length         0.87      -0.43       1.00       0.96
# petal width          0.82      -0.37       0.96       1.00
```

Interpretar Matriz de Correlação

Como ler a matriz:

Diagonal (sempre 1.0):

- Variável correlacionada consigo mesma = perfeito
- Não é interessante

Valores mais altos (perto de ±1):

- **0.96:** Petal length ↔ Petal width (muito forte!)
- **0.87:** Sepal length ↔ Petal length (forte)
- Essas variáveis ”andam juntas”

Valores negativos:

- **-0.43:** Sepal width ↔ Petal length
- Relação inversa (mas fraca)

Valores perto de 0:

- **-0.12:** Sepal length ↔ Sepal width
- Praticamente sem relação

Encontrar Correlações Mais Fortes

Python

```python
# Funcao para encontrar top correlacoes
def top_correlations(corr_matrix, n=5):
    # Pegar triangulo superior (sem diagonal)
    corr_upper = corr_matrix.where(
        np.triu(np.ones(corr_matrix.shape), k=1).astype(bool)
    )
    # Empilhar e ordenar por valor absoluto
    corr_pairs = corr_upper.stack()
    corr_pairs = corr_pairs.reindex(
        corr_pairs.abs().sort_values(ascending=False).index
    )
    return corr_pairs.head(n)
print("=== TOP 5 CORRELACOES ===")
print(top_correlations(corr_matrix))
```

Encontrar Correlações Mais Fortes (cont.)

Python

```
# === TOP 5 CORRELACOES ===
# petal length (cm)  petal width (cm)     0.962865
# sepal length (cm)  petal length (cm)    0.871754
#                    petal width (cm)     0.817941
# sepal width (cm)   petal length (cm)   -0.428440
#                    petal width (cm)    -0.366126
```

Caso de Uso: Features Importantes

Cenário: Prever preço de casas

Correlações encontradas:

- Área total ↔ Preço: **0.85** (forte!)
- Quartos ↔ Preço: **0.62** (moderada)
- Banheiros ↔ Preço: **0.58** (moderada)
- Cor da casa ↔ Preço: **0.05** (nenhuma)

Insights:

1. **Área é o preditor mais forte**
   - Focar em obter medidas precisas de área
2. **Quartos e banheiros importam**
   - Incluir no modelo
3. **Cor não importa**
   - Pode remover do modelo (simplificar)

Exercício: Explorar Correlações

Exercício Prático

Análise de correlações no Iris:

1. Calcule a matriz de correlação
2. Identifique:
   - Qual par tem correlação mais forte?
   - Há correlações negativas?
   - Alguma correlação surpreendente?
3. Use a função top_correlations()
4. Interprete: por que petal length e petal width têm correlação tão alta?
5. Pense: essas variáveis têm relação causal ou apenas correlação?

Dica: Pétalas maiores tendem a ser proporcionalmente mais largas

Resumo: Correlação

Conceitos-chave:

- **Correlação:** Mede relação linear entre variáveis
- **Escala:** -1 (negativa perfeita) a +1 (positiva perfeita)
- **Interpretação:**
  - $|r| > 0.7$: forte
  - $0.3 < |r| < 0.7$: moderada
  - $|r| < 0.3$: fraca
- **Correlação ≠ Causalidade:** SEMPRE lembrar disso!
- **Matriz:** Ver todas as correlações de uma vez

Ferramentas Pandas:

- `.corr()`: calcular correlação
- Matriz: `df[cols].corr()`

Scatter Plot: O que é?

Scatter plot (gráfico de dispersão) mostra relação entre duas variáveis numéricas

Como funciona:

- Cada ponto = uma observação
- Posição X = valor da primeira variável
- Posição Y = valor da segunda variável
- Padrão dos pontos revela a relação

O que revela:

- Direção da relação (positiva/negativa)
- Força da relação (pontos concentrados ou dispersos)
- Outliers (pontos muito distantes)
- Padrões não-lineares

Scatter Plot: O que é?

⚠ Atenção

Scatter plot é a MELHOR forma de visualizar correlação!

O que Scatter Plots Revelam

Informações visuais imediatas:

1. Direção:

- Pontos sobem da esquerda $\rightarrow$ direita = correlação positiva
- Pontos descem da esquerda $\rightarrow$ direita = correlação negativa
- Pontos espalhados = sem correlação

2. Força:

- Pontos próximos a uma linha = correlação forte
- Pontos muito dispersos = correlação fraca

O que Scatter Plots Revelam

Informações visuais imediatas (cont.):

3. Forma:

- Linear (reta)
- Curva (não-linear)
- Sem padrão (aleatório)

4. Anomalias:

- Outliers ficam visíveis
- Grupos/clusters separados

Criar Scatter Plot Básico

Python

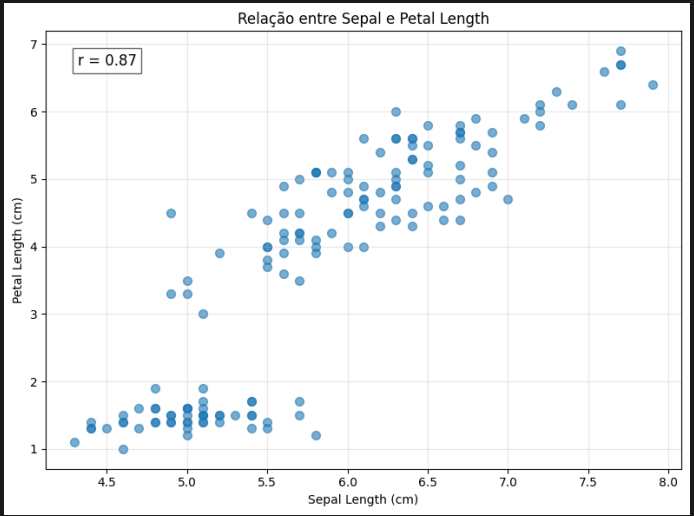
```python
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
# Calcular o coeficiente de correlação de Pearson
r, _ = pearsonr(df['sepal length (cm)'], df['petal length (cm)'])
# Scatter plot simples
fig, ax = plt.subplots(figsize=(8, 6))
ax.scatter(df['sepal length (cm)'],
           df['petal length (cm)'],
           alpha=0.6,    # Transparencia
           s=50)         # Tamanho dos pontos
ax.set_xlabel('Sepal Length (cm)')
ax.set_ylabel('Petal Length (cm)')
ax.set_title('Relacao entre Sepal e Petal Length')
ax.grid(True, alpha=0.3)
```

Criar Scatter Plot Básico (cont.)

Python

```python
# Exibir o valor de r diretamente no gráfico
ax.text(0.05, 0.95, f'r = {r:.2f}',
        transform=ax.transAxes,
        fontsize=12,
        verticalalignment='top',
        bbox=dict(facecolor='white', alpha=0.6))

plt.tight_layout()
plt.show()
```

Criar Scatter Plot Básico (cont.)



Identificar Correlações Visualmente

Padrões visuais vs valores de correlação:

r ≈ +0.9 (forte positiva)

Pontos formam linha crescente clara
Pouca dispersão
Fácil prever Y por X

r ≈ +0.5 (moderada)

Tendência visível mas dispersa
Muita variabilidade

r ≈ -0.9 (forte negativa)

Pontos formam linha decrescente clara
Pouca dispersão
Relação inversa evidente

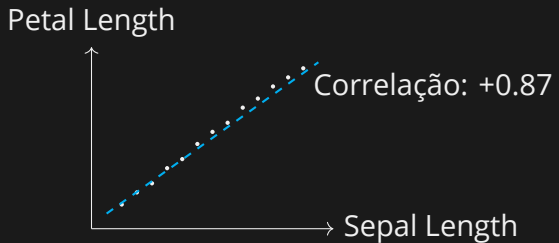
r ≈ 0 (nenhuma)

Nuvem de pontos aleatória
Sem padrão discernível

Correlação Positiva no Scatter

Exemplo: Sepal Length vs Petal Length (Iris)

O que você vê:



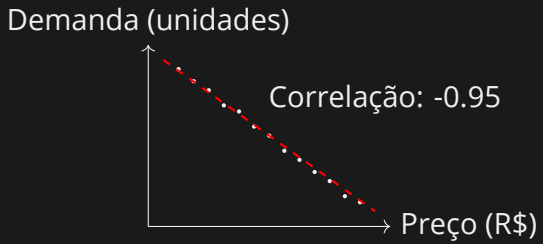
Interpretação:

- Flores com sépalas maiores têm pétalas maiores
- Relação forte e consistente
- Podemos prever petal length conhecendo sepal length

Correlação Negativa no Scatter

Exemplo hipotético: Preço vs Demanda

O que você veria:



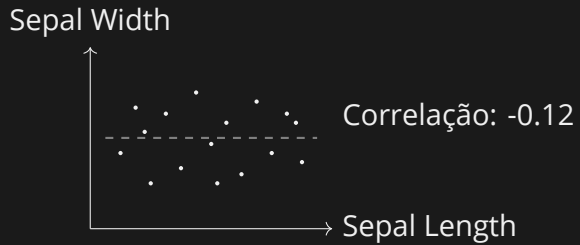
Interpretação:

- Quanto maior o preço, menor a demanda
- Relação inversa forte
- Lei da oferta e demanda visual!

Sem Correlação no Scatter

Exemplo: Sepal Width vs Sepal Length (Iris)

O que você veria:



Interpretação:

- Nenhum padrão claro
- Comprimento não prediz largura
- Variáveis independentes

Scatter Plot Colorido por Categoria

</> Python

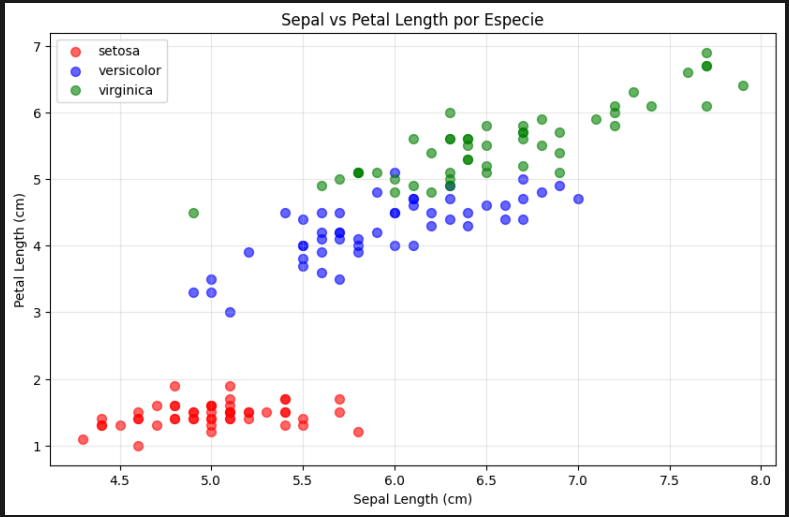
```python
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 6))

# Cores para cada especie
colors = {'setosa': 'red',
          'versicolor': 'blue',
          'virginica': 'green'}

# Plot cada especie com cor diferente
for species, color in colors.items():
    subset = df[df['species'] == species]
    ax.scatter(subset['sepal length (cm)'],
               subset['petal length (cm)'],
               c=color, label=species,
               alpha=0.6, s=50)
```

Scatter Plot Colorido por Categoria (cont.)

Python

```python
ax.set_xlabel('Sepal Length (cm)')
ax.set_ylabel('Petal Length (cm)')
ax.set_title('Sepal vs Petal Length por Especie')
ax.legend()
ax.grid(True, alpha=0.3)
plt.show()
```

Scatter Plot Colorido por Categoria (cont.)



Tamanho e Transparência dos Pontos

Customizar aparência para melhor visualização:

1. Tamanho (s):

- `s=10`: Pontos pequenos (muitos dados)
- `s=50`: Tamanho padrão
- `s=100`: Pontos grandes (poucos dados, ênfase)
- Pode variar por variável: `s=df['column']*10`

2. Transparência (alpha):

- `alpha=1.0`: Totalmente opaco
- `alpha=0.6`: Recomendado (ver sobreposições)
- `alpha=0.3`: Muito transparente (muitos pontos)
- Útil para ver densidade: áreas mais escuras = mais pontos

Tamanho e Transparência dos Pontos

Nota Importante

Use alpha < 1.0 quando houver sobreposição de pontos

Adicionar Linha de Tendência (Conceito)

</> Python

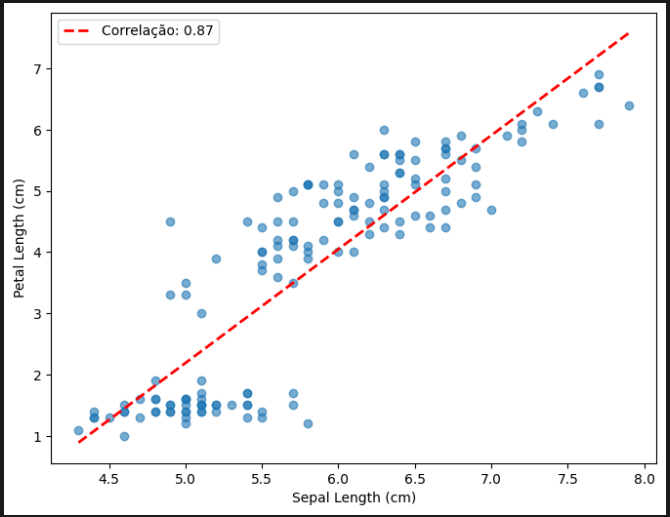
```python
from scipy import stats

fig, ax = plt.subplots(figsize=(8, 6))

x = df['sepal length (cm)']
y = df['petal length (cm)']

# Scatter plot
ax.scatter(x, y, alpha=0.6)

# Calcular linha de tendencia (regressao linear)
slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
line_x = np.array([x.min(), x.max()])
line_y = slope * line_x + intercept
```

Adicionar Linha de Tendência (Conceito) (cont.)

</> Python

```python
# Plotar linha
ax.plot(line_x, line_y, 'r--', linewidth=2,
        label=f'Correlação: {r_value:.2f}')

ax.legend()
ax.set_xlabel('Sepal Length (cm)')
ax.set_ylabel('Petal Length (cm)')
plt.show()
```

Adicionar Linha de Tendência (Conceito) (cont.)



Caso de Uso: Identificar Segmentos

Cenário: Análise de clientes (gastos vs visitas)

Scatter plot revela 3 grupos:

1. **Alto valor (canto superior direito):**
   - Muitas visitas + altos gastos
   - VIPs - recompensar com programa de fidelidade
2. **Visitantes frequentes, baixo gasto (esquerda superior):**
   - Muitas visitas mas gastam pouco
   - Oportunidade: oferecer upsell
3. **Ocasionais alto valor (direita inferior):**
   - Poucas visitas mas gastam muito
   - Estratégia: aumentar frequência de visitas

Ação: Cada grupo recebe estratégia diferente!

Exercício: Explorar Relações

✋ Exercício Prático

Crie scatter plots e analise:

1. Crie scatter plot de `petal length` vs `petal width`
2. Colorir por espécie
3. Responda:
   - A correlação é positiva ou negativa?
   - É forte, moderada ou fraca?
   - As espécies formam grupos separados?
   - Há outliers visíveis?
4. Compare com scatter de `sepal length` vs `sepal width`
5. Qual par de variáveis tem relação mais clara?

Dica: Use o código do slide anterior como base

Resumo: Scatter Plots

Pontos-chave:

- **Melhor forma** de visualizar correlação
- **Cada ponto** = uma observação
- **Padrão dos pontos** revela:
  - Direção (positiva/negativa/nenhuma)
  - Força (concentrado/disperso)
  - Outliers (pontos isolados)
  - Grupos/clusters
- **Customizações úteis:**
  - Cores por categoria
  - Transparência para sobreposição
  - Linha de tendência

⚠ Atenção

Sempre crie scatter plot ANTES de confiar só na correlação numérica!

Outliers: Revisão e Aprofundamento

Recapitulando da Aula 07:

- Outliers = valores anormalmente distantes
- Regra IQR: $< Q1 - 1.5 \times IQR$ ou $> Q3 + 1.5 \times IQR$
- Visíveis em boxplots

Hoje: mais contexto e decisões

- Por que outliers importam tanto?
- Como identificar em múltiplas dimensões?
- O que fazer quando encontrar outliers?

Por que Outliers Importam?

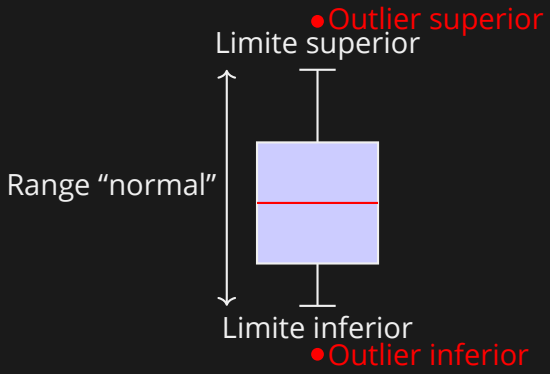
Impactos negativos:

1. **Distorcem estatísticas:**
   - Média puxada para extremo
   - Desvio padrão inflado
   - Correlação afetada
2. **Afetam modelos:**
   - Modelos tentam ”explicar”outliers
   - Performance geral piora
   - Previsões erradas
3. **Podem esconder padrões:**
   - Padrões do grupo majoritário ficam invisíveis
   - Visualizações distorcidas (escala muito grande)

MAS: Nem todo outlier deve ser removido!

Identificar Outliers em Boxplots

Revisão visual - Boxplot mostra:



Pontos além dos whiskers = outliers

Identificar Outliers em Scatter Plots

Scatter plots revelam outliers em 2 dimensões:

</> Python

```python
fig, ax = plt.subplots(figsize=(8, 6))
ax.scatter(df['sepal length (cm)'],
           df['petal length (cm)'],
           alpha=0.6)
ax.set_xlabel('Sepal Length (cm)')
ax.set_ylabel('Petal Length (cm)')
ax.set_title('Identificar Outliers em 2D')
# Outliers aparecem como pontos isolados,
# longe da nuvem principal de pontos
plt.show()
# Procure por pontos que estao:
# - Muito acima/abaixo da tendencia principal
# - Isolados no canto do grafico
# - Distantes do cluster principal
```

O que Fazer com Outliers?

Guia de decisão:

1. Investigar primeiro (SEMPRE):

- É um erro de medição/digitação? $\rightarrow$ Corrigir ou remover
- É um caso genuíno raro? $\rightarrow$ Manter ou tratar separadamente
- Faz sentido no contexto? $\rightarrow$ Analisar causa

2. Opções de tratamento:

- **Remover:** Se for erro óbvio
- **Substituir:** Por média/mediana do grupo
- **Transformar:** Log ou outras transformações
- **Segmentar:** Analisar outliers separadamente
- **Manter:** Se forem casos legítimos importantes

O que Fazer com Outliers?

⚠ Atenção

NUNCA remova outliers sem investigar! Podem ser seus dados mais valiosos.

Exemplos: Outliers Bons vs Ruins

Outliers que DEVEM ser removidos:

- Idade = 200 anos (erro de digitação)
- Salário = -R$ 5.000 (impossível)
- Altura = 5 metros (erro de unidade)

Outliers que podem ser MANTIDOS:

- Cliente que gasta 10x a média (VIP real)
- Casa de R$ 10 milhões (mansão legítima)
- Produto com 1000x vendas (viral hit)

Regra:

- Se é **fisicamente impossível** $\rightarrow$ remover
- Se é **raro mas possível** $\rightarrow$ manter ou analisar separadamente
- Se é **informativo** (fraude, evento especial) $\rightarrow$ DEFINITIVAMENTE manter!

Caso de Uso: Detecção de Anomalias

Cenário: Sistema de detecção de fraudes em cartões

Transações normais:

- R$ 20-200 em supermercados
- 2-5 transações por dia
- Sempre na mesma cidade

Outlier detectado:

- Transação de R$ 5.000
- 20 transações em 1 hora
- Em cidade diferente

Ação:

- Sistema bloqueia cartão
- Liga para cliente
- **Outlier SALVA dinheiro do cliente!**

Caso de Uso: Detecção de Anomalias

Cenário: Sistema de detecção de fraudes em cartões

⚠ Atenção

Neste caso, outliers são EXATAMENTE o que queremos encontrar!

Exercício: Análise de Outliers

Exercício Prático

Investigue outliers no Iris:

1. Use a função detect_outliers_iqr() da Aula 07:

```
for col in numeric_cols:
    outliers, lower, upper = detect_outliers_iqr(df[col])
    if len(outliers) > 0:
        print(f"\n{col}:")
        print(f"Valores: {outliers.values}")
        print(f"Especies: {df.loc[outliers.index, 'species'].values}")
```

Exercício: Análise de Outliers

✋ Exercício Prático

Investigue outliers no Iris:

1. Use a função detect_outliers_iqr() da Aula 07 (cont.)
2. Crie boxplot mostrando outliers
3. Perguntas:
   - Os outliers são de alguma espécie específica?
   - Fazem sentido biologicamente?
   - Devemos removê-los?

Resumo: Outliers

Pontos-chave:

- **Identificar:** IQR method, boxplots, scatter plots
- **Investigar:** Sempre pergunte ”por quê?”
- **Decisão:**
  - Erro → remover/corrigir
  - Raro mas legítimo → manter ou analisar separado
  - Informativo → MANTER!
- **Impacto:** Distorcem estatísticas e modelos
- **Contexto:** Outliers podem ser seus dados mais valiosos

Nota Importante

Documente SEMPRE sua decisão sobre outliers: ”Removi X porque Y”

Bloco 2

Seaborn - Visualizações Avançadas

Seaborn: O que é?

Seaborn = Matplotlib turbinado

Vantagens:

- **Menos código:** 1 linha faz o que Matplotlib faz em 10
- **Mais bonito:** Cores e estilos modernos por padrão
- **Integrado:** Funciona direto com DataFrames Pandas
- **Estatístico:** Visualizações pensadas para análise de dados
- **Temas:** Estilos profissionais prontos

Quando usar:

- **Seaborn:** EDA, apresentações, relatórios
- **Matplotlib:** Quando precisa controle total e customização

Nota Importante

Seaborn é construído SOBRE Matplotlib - você pode misturar os dois!

Seaborn *vs* Matplotlib

Matplotlib:

- Mais código
- Controle fino
- Flexibilidade total
- Aparência básica
- Ideal para gráficos customizados

Código:

- 10-15 linhas
- Manual

Seaborn:

- Menos código
- Alto nível
- Padrões inteligentes
- Bonito por padrão
- Ideal para EDA rápida

Código:

- 1-3 linhas
- Automático

Analogia:

Matplotlib = Carro manual | Seaborn = Carro automático

Importar e Configurar Seaborn

Python

```python
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Configuracao do estilo
sns.set_theme()  # Ativa tema Seaborn

# Ou escolher estilo especifico:
# sns.set_style("darkgrid")  # Grade escura
# sns.set_style("whitegrid")  # Grade clara
# sns.set_style("dark")       # Sem grade, fundo escuro
# sns.set_style("white")      # Sem grade, fundo claro
# sns.set_style("ticks")      # Com marcas nos eixos
```

Importar e Configurar Seaborn (cont.)

Python

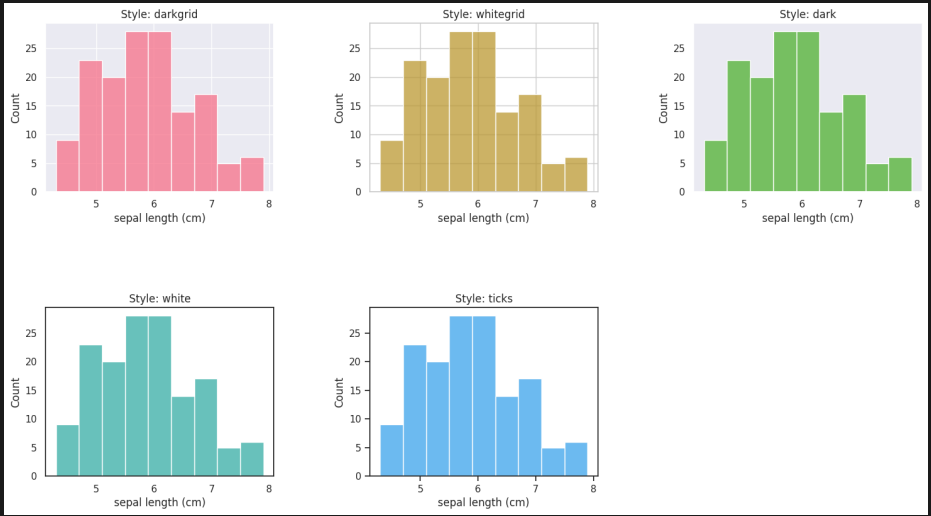
```python
# Paleta de cores
sns.set_palette("husl")  # Cores vibrantes
# Outras: "deep", "muted", "bright", "pastel", "dark"

# Tamanho de fonte
sns.set_context("notebook")  # Padrao
# Outras: "paper", "talk", "poster"
```

Temas e Estilos

Python

```python
# Comparar diferentes estilos
styles = ['darkgrid', 'whitegrid', 'dark', 'white', 'ticks']
fig = plt.figure(figsize=(15, 8))
subfigs = fig.subfigures(2, 3, wspace=0.1, hspace=0.3)
for idx, subfig in enumerate(subfigs.flat):
    if idx >= len(styles):
       subfig.set_visible(False)
       continue
    style = styles[idx]
    with sns.axes_style(style):
        ax = subfig.add_subplot(1,1,1)
        sns.histplot(data=df, x='sepal length (cm)', ax=ax, color=sns.color_palette("husl")[idx])
        ax.set_title(f'Style: {style}')
plt.show()
```

Temas e Estilos (cont.)



Temas e Estilos (cont.)

- Para EDA: *darkgrid* ou *whitegrid*
  - *darkgrid*: ideal para visualizações exploratórias em notebooks, pois o fundo acinzentado destaca padrões, especialmente em séries e dispersões densas.
  - *whitegrid*: excelente para gráficos baseados em categorias (como boxplots e barplots); o contraste branco facilita comparações.

- Para apresentações: *white* ou *ticks*
  - *white*: aparência limpa e neutra; destaca melhor o conteúdo quando usado em slides ou publicações.
  - *ticks*: variação mais formal, com eixos bem definidos. Recomendada para artigos, relatórios técnicos, publicações acadêmicas.

Temas e Estilos (cont.)

- Para dashboards e temas escuros: *dark*
  - Útil em contextos com fundo escuro ou interfaces digitais noturnas; reduz fadiga visual e se integra bem a paletas vivas.
- Para publicações impressas: *whitegrid* ou *ticks*
  - Ambos garantem boa visibilidade em tons de cinza e excelente legibilidade em documentos em papel.
- Para visualização rápida em ambientes de código (como Jupyter/Colab): *darkgrid*
  - É o estilo padrão do Seaborn, balanceando contraste e legibilidade, o que o torna prático para prototipagem.

Integração com Pandas

Seaborn trabalha direto com DataFrames:

Matplotlib:

- Passa arrays/listas
- `plt.scatter(x, y)`
- Precisa extrair dados

Seaborn:

- Passa DataFrame
- `sns.scatterplot(data=df, x='col1', y='col2')`
- Trabalha direto com nomes

Vantagem:

- Código mais limpo e legível
- Menos bugs (nomes em vez de índices)
- Fácil adicionar categorias (`hue='species'`)

Distribution Plots

Visualizar distribuições de forma moderna:

Função	O que faz	Quando usar
histplot	Histograma moderno	Distribuição básica
kdeplot	Curva de densidade	Distribuição suave
displot	Versátil (hist+kde)	Exploração geral
rugplot	Marcas individuais	Ver pontos exatos

Todos são melhores que os equivalentes Matplotlib:

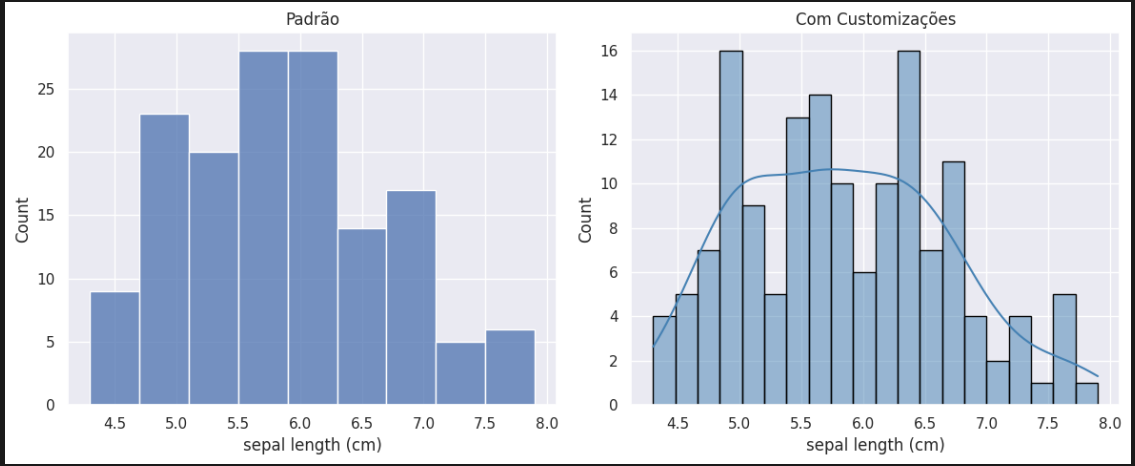
- Mais bonitos
- Menos código
- Mais opções automáticas

histplot: Histograma Moderno

</> Python

```python
# Histograma simples
sns.histplot(data=df, x='sepal length (cm)')
plt.show()
# Com customizacoes
sns.histplot(data=df,
             x='sepal length (cm)',
             bins=20,
             kde=True,     # Adiciona curva de densidade
             color='steelblue',
             edgecolor='black')
plt.title('Distribuicao de Sepal Length')
plt.show()
# Compare com Matplotlib:
# - Menos linhas de codigo, mais bonito por padrao
# - KDE em uma opção!
```

histplot: Histograma Moderno (cont.)



KDE: O que é?

KDE = Kernel Density Estimation (Estimativa de Densidade por Kernel)

Conceito simples:

- Imagine o histograma como blocos
- KDE ”suaviza”esses blocos em uma curva
- Mostra a forma da distribuição sem ”degraus”

Analogia:

Histograma = foto pixelada
KDE = foto suavizada

Vantagens:

- Mais fácil ver a forma geral
- Sem dependência do número de bins
- Visual mais profissional

KDE: O que é?

Nota Importante

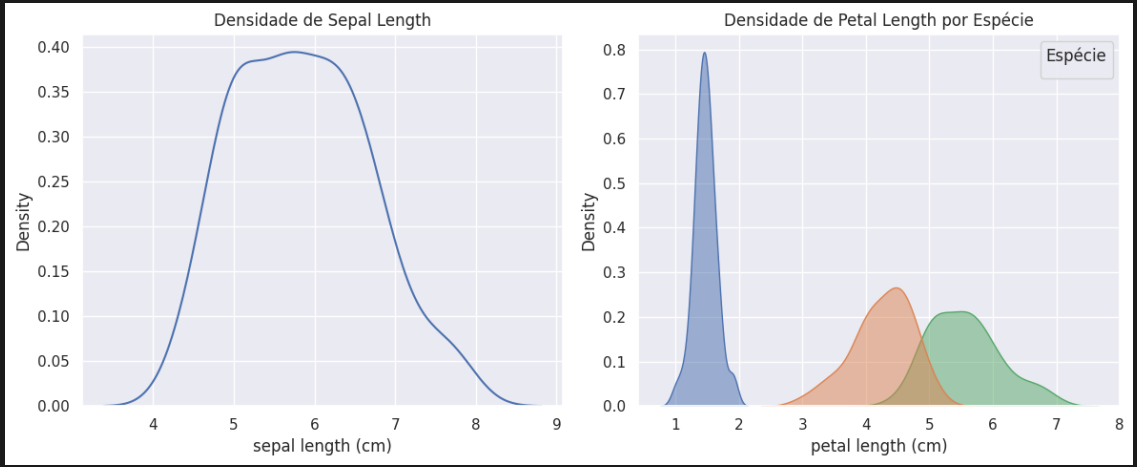
Não se preocupe com a matemática - use KDE para ver distribuições suaves!

kdeplot: Curva de Densidade

Python

```python
# KDE puro (sem histograma)
sns.kdeplot(data=df, x='sepal length (cm)')
plt.title('Densidade de Sepal Length')
plt.show()

# Comparar multiplas distribuicoes
sns.kdeplot(data=df, x='petal length (cm)',
            hue='species', fill=True, alpha=0.5)
plt.title('Densidade de Petal Length por Especie')
plt.legend(title='Species')
plt.show()

# fill=True: preenche area sob a curva
# alpha=0.5: transparência para ver sobreposições
# hue='species': uma curva para cada espécie
```

kdeplot: Curva de Densidade (cont.)

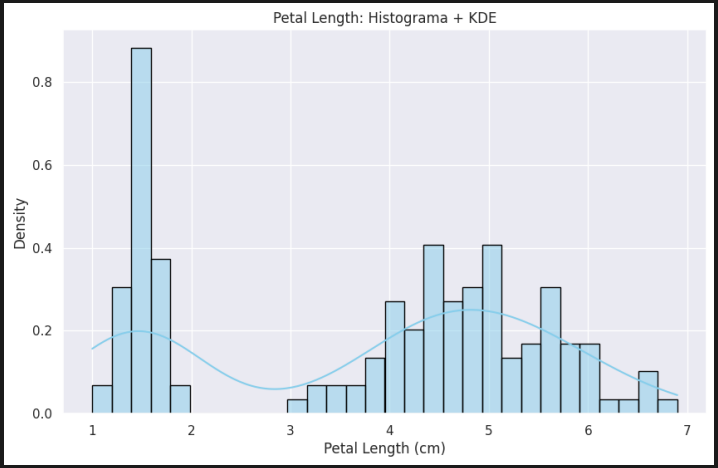


Histograma + KDE Juntos

Python

```python
# Melhor dos dois mundos
fig, ax = plt.subplots(figsize=(10, 6))
sns.histplot(data=df,
             x='petal length (cm)',
             bins=30,
             kde=True,        # Adiciona KDE
             color='skyblue',
             edgecolor='black',
             stat='density')  # Normaliza para densidade
plt.title('Petal Length: Histograma + KDE')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Density')
plt.show()
# stat='density': faz histograma e KDE comparaveis
```

Histograma + KDE Juntos (cont.)

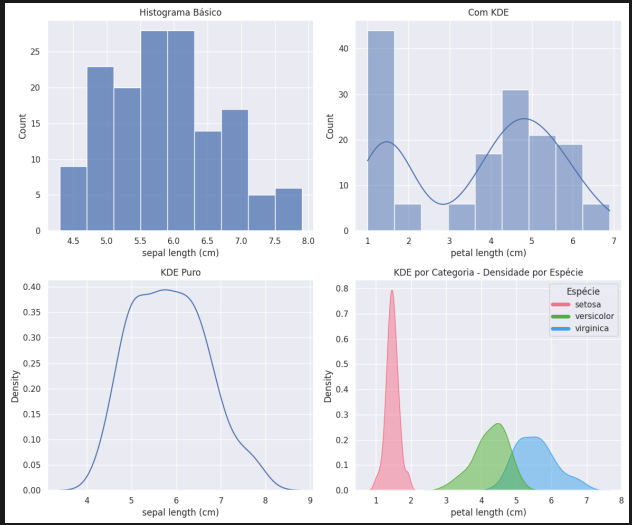


displot: Distribuições Versáteis

Python

```python
# displot = histplot + kdeplot + mais opcoes
# Recomendado para exploracao inicial
# Histograma basico
sns.displot(data=df, x='sepal length (cm)',
            height=5, aspect=1.5)
plt.show()
# Com KDE
sns.displot(data=df, x='petal length (cm)',
            kde=True, height=5, aspect=1.5)
plt.show()
# KDE Por categoria (facetas - mostra densidade por espécie)
sns.displot(data=df, x='petal length (cm)',
            hue='species', kde=True,
            height=5, aspect=1.5)
plt.show()
```

displot: Distribuições Versáteis (cont.)

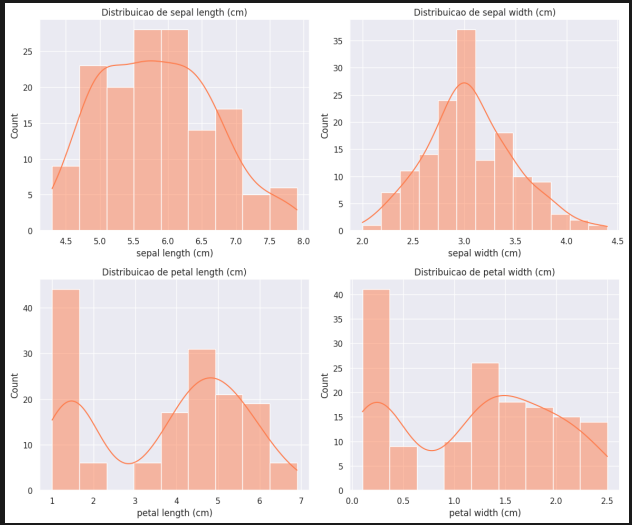


Comparar Distribuições Lado a Lado

</> Python

```python
# Criar subplots de distribuicoes
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes = axes.flatten()
cols = ['sepal length (cm)', 'sepal width (cm)',
        'petal length (cm)', 'petal width (cm)']
for ax, col in zip(axes, cols):
    sns.histplot(data=df, x=col, kde=True,
                 color='coral', ax=ax)
    ax.set_title(f'Distribuicao de {col}')
plt.tight_layout()
plt.show()
```

Comparar Distribuições Lado a Lado (cont.)

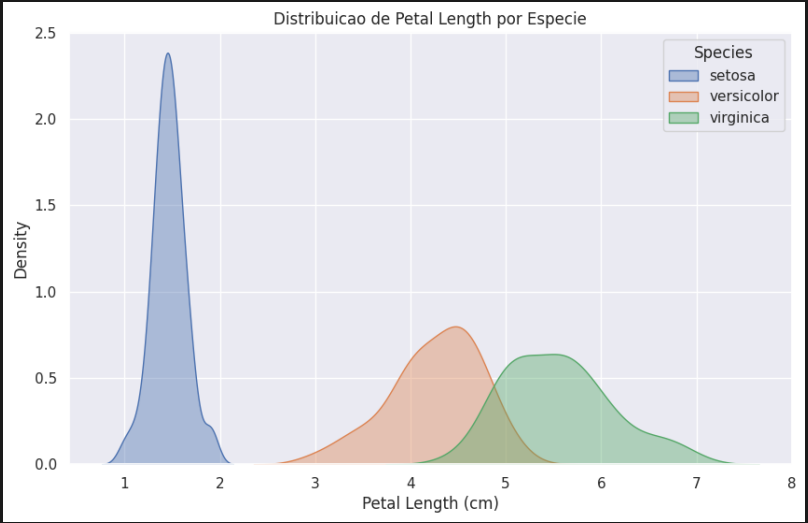


Múltiplas Distribuições Sobrepostas

Python

```python
# Comparar especies na mesma figura
plt.figure(figsize=(10, 6))
for species in df['species'].unique():
    subset = df[df['species'] == species]
    sns.kdeplot(data=subset, x='petal length (cm)',
                label=species, fill=True, alpha=0.4)
plt.title('Distribuicao de Petal Length por Especie')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Density')
plt.legend(title='Species')
plt.show()
# Visualiza claramente:
# - Setosa: pico a esquerda
# - Virginica: pico a direita
# - Versicolor: no meio
```

Múltiplas Distribuições Sobrepostas (cont.)



Caso de Uso: Comparar Antes/Depois

Cenário: Avaliar impacto de intervenção

Exemplo - Tempo de resposta de site:

- **Antes da otimização:**
  - Distribuição larga (0.5s a 5s)
  - Média alta
  - Muitos outliers lentos
- **Depois da otimização:**
  - Distribuição estreita (0.2s a 1s)
  - Média baixa
  - Poucos outliers

KDE sobreposto mostra claramente:

- Deslocamento do pico (média melhorou)
- Estreitamento (consistência melhorou)
- Redução de cauda (menos casos extremos)

Exercício: Distribuições com Seaborn

Exercício Prático

Explore distribuições:

1. Crie histplot de `sepal width` com KDE
2. Crie kdeplot comparando as 3 espécies para `petal width`
3. Use displot com `hue='species'` para `sepal length`
4. Responda:
   - Qual variável tem distribuição mais simétrica?
   - Qual variável melhor separa as espécies?
   - Alguma espécie tem distribuição bimodal?

Dica: Compare formas e posições dos picos

Resumo: Distribution Plots

Principais funções:

- **histplot():** Histograma moderno + KDE opcional
- **kdeplot():** Curva de densidade suave
- **displot():** Versátil, recomendado para EDA

Vantagens sobre Matplotlib:

- 1-2 linhas vs 5-10 linhas
- KDE integrado
- Cores e estilos melhores
- Fácil comparar categorias (hue)

Quando usar:

- Ver forma da distribuição
- Comparar grupos
- Identificar modas
- EDA rápida

Categorical Plots

Visualizações especializadas para categorias:

Função	O que mostra	Similar a
countplot	Frequências	Bar chart
boxplot	Quartis + outliers	Matplotlib boxplot
violinplot	Boxplot + densidade	Novo!
stripplot	Pontos individuais	Scatter 1D
swarmplot	Pontos organizados	Stripplot melhor
barplot	Médias + IC	Bar chart com erro

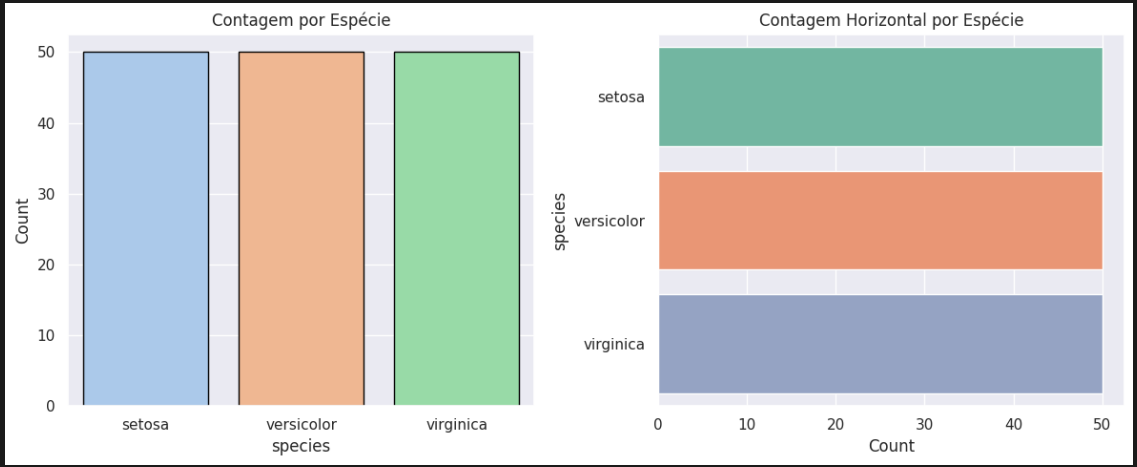
Todos trabalham com categorias no eixo X ou Y

countplot: Frequências Elegantes

Python

```python
# Contar frequencias automaticamente
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='species',
              palette='pastel', edgecolor='black')
plt.title('Contagem por Especie')
plt.ylabel('Count')
plt.show()
# Compare com Matplotlib bar chart:
# - Nao precisa fazer value_counts() manualmente
# - Cores bonitas automaticas
# - Menos codigo

# Horizontal
sns.countplot(data=df, y='species', palette='Set2')
plt.show()
```

countplot: Frequências Elegantes (cont.)

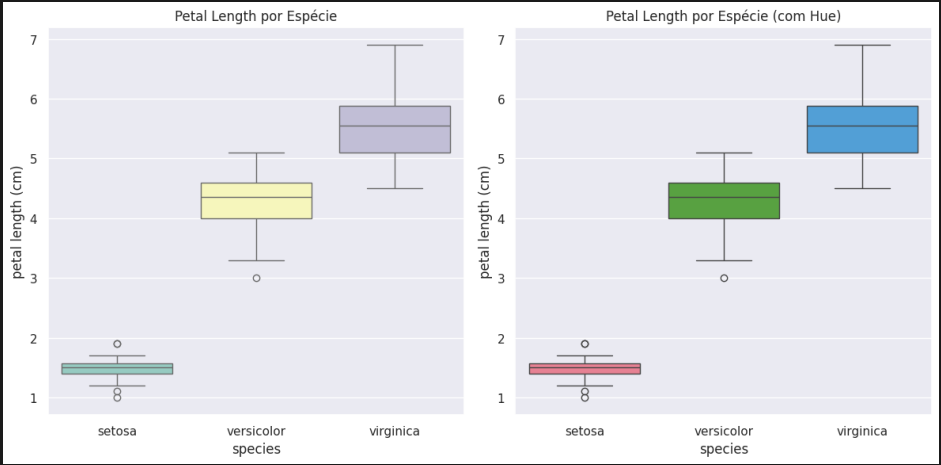


boxplot: Versão Seaborn

Python

```python
# Boxplot por categoria
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='species', y='petal length (cm)',
            palette='Set3')
plt.title('Petal Length por Especie')
plt.show()
# Vantagens sobre Matplotlib:
# - Sintaxe mais simples
# - Cores bonitas automaticas
# - Facil adicionar mais dimensoes
# Com mais uma variavel (hue)
sns.boxplot(data=df, x='species', y='petal length (cm)',
            hue='species', palette='husl', legend=False)
plt.show()
```

boxplot: Versão Seaborn (cont.)



violinplot: Boxplot + Distribuição

Violin plot = Boxplot + KDE

O que mostra:

- **Largura:** Densidade (como KDE)
  - Mais largo = mais valores naquela altura
- **Linha branca central:** Quartis (como boxplot)
- **Formato:** Forma da distribuição

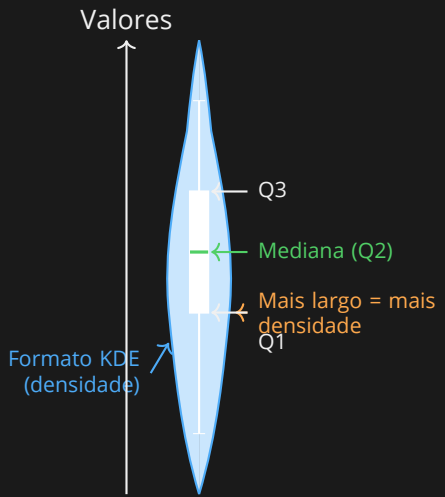
Vantagens:

- Mais informação que boxplot
- Vê se distribuição é bimodal
- Mostra assimetria claramente

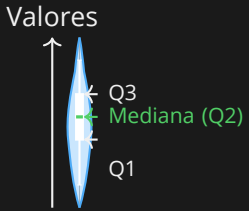
Quando usar:

- Comparar distribuições entre grupos
- Ver forma completa (não só quartis)
- Apresentações (visual interessante)

Anatomia do Violin Plot



Anatomia do Violin Plot (cont.)



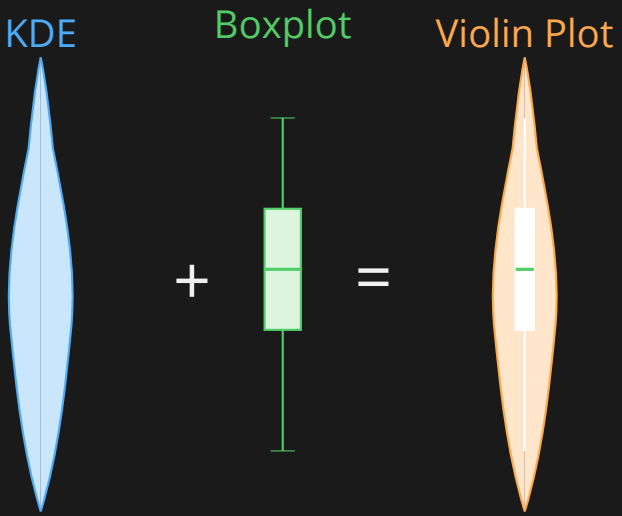
Forma = Distribuição

- Simétrica: distribuição normal
- Bimodal: dois picos
- Assimétrica: enviesada

Centro = Estatísticas

- Caixa branca: boxplot
- Linha verde: mediana
- Whiskers: limites

Violin Plot = Boxplot + KDE



violinplot: Código

Python

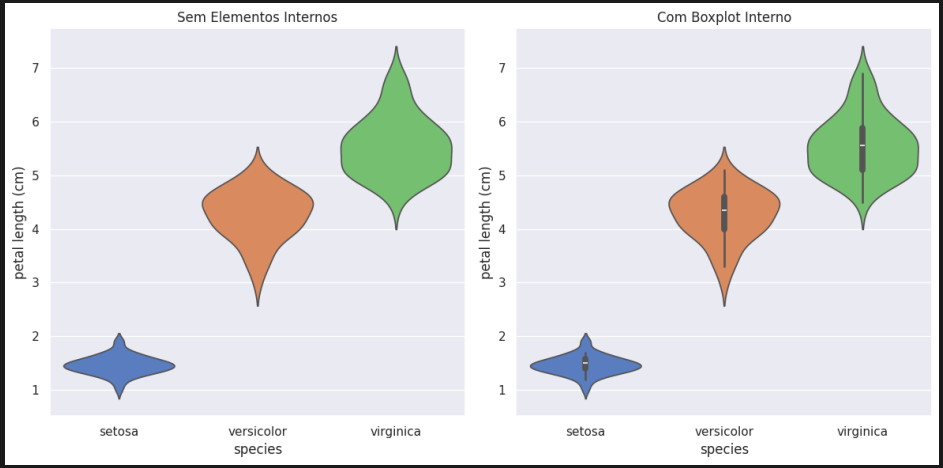
```python
fig, axes = plt.subplots(1, 2, figsize=(12, 6))
# Violin plot sem elementos internos
sns.violinplot(
    data=df,
    x='species',
    y='petal length (cm)',
    hue='species',
    palette='muted',
    inner=None,      # remove boxplot interno
    ax=axes[0]
)
axes[0].set_title('Sem Elementos Internos')
```

violinplot: Código

Python

```python
# Violin plot com boxplot interno (padrão)
sns.violinplot(
    data=df,
    x='species',
    y='petal length (cm)',
    hue='species',
    palette='muted',
    inner='box',
    ax=axes[1]
)
axes[1].set_title('Com Boxplot Interno')

plt.tight_layout()
plt.show()
```

violinplot: Código (cont.)

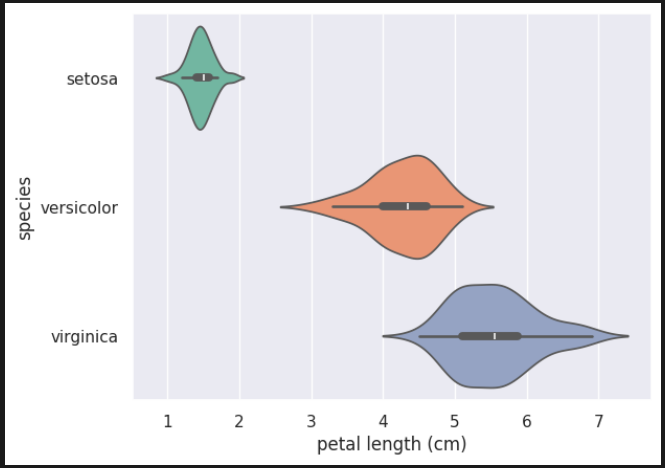


violinplot Horizontal: Código

Python

```python
# Violinplot horizontal
sns.violinplot(data=df, x='petal length (cm)',
               y='species', palette='Set2') #Só invertemos os eixos
plt.show() # Boxplot é padrão

```

violinplot Horizontal: Código (cont.)

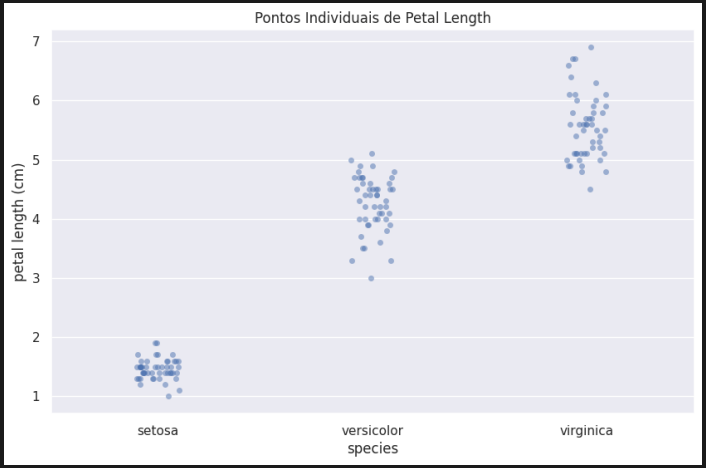


stripplot: Ver Pontos Individuais

</> Python

```python
# Mostrar cada ponto individual
plt.figure(figsize=(10, 6))
sns.stripplot(data=df, x='species',
              y='petal length (cm)',
              alpha=0.5)  # Transparencia para sobreposicao
plt.title('Pontos Individuais de Petal Length')
plt.show()
# Util para:
# - Ver distribuicao exata dos pontos
# - Identificar outliers individuais
# - Datasets pequenos (< 200 pontos)

# Problema: pontos se sobrepoem
# Solução: swarmplot!
```

stripplot: Ver Pontos Individuais (cont.)

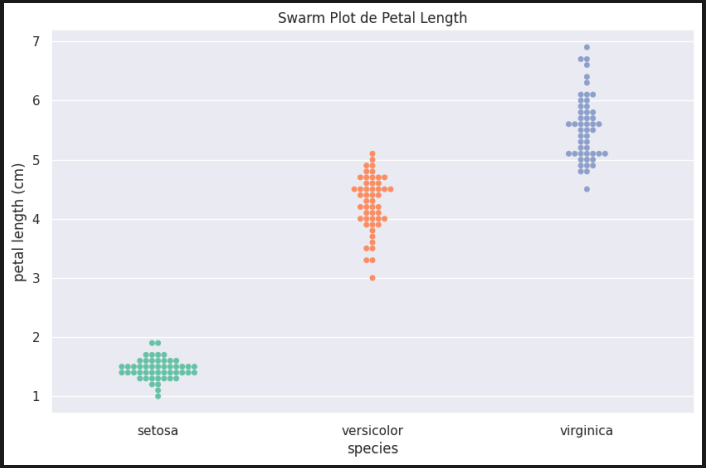


swarmplot: Stripplot Organizado

Python

```python
# Pontos organizados (sem sobreposicao)
plt.figure(figsize=(10, 6))
sns.swarmplot(data=df, x='species',
              y='petal length (cm)',
              hue='species',
              palette='Set2')
plt.title('Swarm Plot de Petal Length')
plt.show()
```

swarmplot: Stripplot Organizado (cont.)



swarmplot: Stripplot Organizado

Vantagens:

- Nenhuma sobreposição
- Vê cada ponto claramente
- Forma da distribuição visível

Desvantagem:

- Lento para muitos pontos (>1000)

Nota Importante

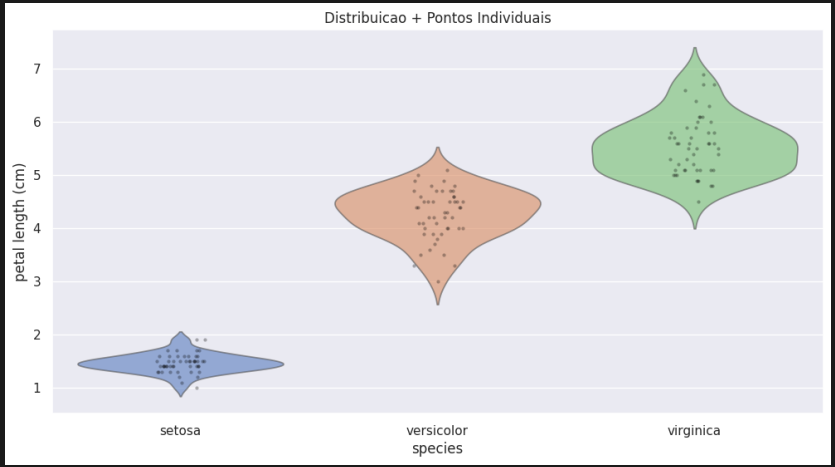
MELHOR: Combinar violin + swarm!

Combinar Plots: Violin + Strip

Python

```python
# Combinacao poderosa: distribuicao + pontos
plt.figure(figsize=(12, 6))
# Primeiro: violin plot
sns.violinplot(data=df, x='species',
               y='petal length (cm)',
               hue='species', palette='muted', alpha=0.6,
               inner=None)  # Sem boxplot interno

# Depois: strip plot em cima
sns.stripplot(data=df, x='species',
              y='petal length (cm)',
              color='black', alpha=0.3, size=3)
plt.title('Distribuicao + Pontos Individuais')
plt.show()
```

Combinar Plots: Violin + Strip (cont.)

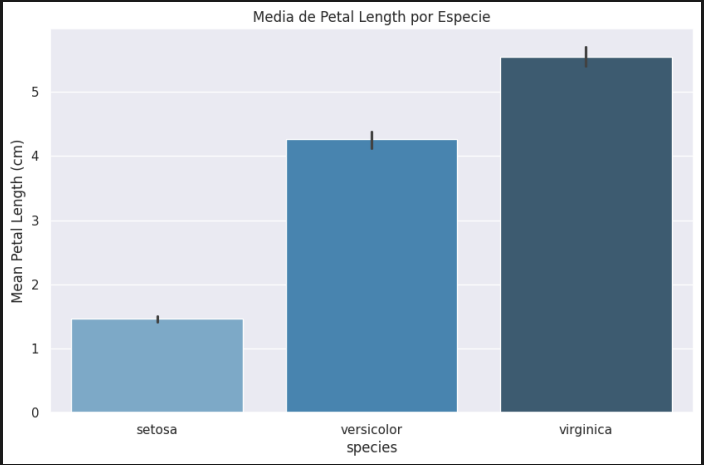


barplot: Médias com Intervalo de Confiança

Python

```python
# Bar plot com media + IC automatico
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='species',
            y='petal length (cm)',
            hue='species',
            palette='Blues_d',
            errorbar='ci')  # Intervalo de confianca (95%)
plt.title('Media de Petal Length por Especie')
plt.ylabel('Mean Petal Length (cm)')
plt.show()
# Barras = medias
# Linhas pretas = intervalo de confianca (incerteza)

# Se IC é pequeno: média é confiavel
# Se IC é grande: muita variabilidade
```

barplot: Médias com Intervalo de Confiança (cont.)



Caso de Uso: A/B Testing Visual

Cenário: Comparar duas versões de site

Métricas:

- Tempo no site (segundos)
- Versão A vs Versão B
- 500 usuários em cada

Violinplot revela:

- **Versão A:** Distribuição ampla (50-200s)
  - Alguns saem rápido, outros ficam muito
  - Inconsistente
- **Versão B:** Distribuição concentrada (80-120s)
  - Maioria fica tempo similar
  - Mais previsível

Decisão: Versão B é melhor (mais consistente)

Caso de Uso: A/B Testing Visual

⚠ Atenção

Violin plot mostra distribuição completa!

Exercício: Categorical Plots

Exercício Prático

Compare espécies com diferentes plots:

1. Crie countplot das espécies
2. Crie boxplot de `sepal width` por espécie
3. Crie violinplot de `petal width` por espécie
4. Combine violin + swarm para `sepal length`
5. Responda:
   - Qual plot mostra mais informação?
   - Alguma espécie tem distribuição bimodal?
   - Os intervalos de confiança se sobrepõem?

Dica: Violin plots revelam formas que boxplots escondem

Resumo: Categorical Plots

Principais funções:

Plot	Melhor uso
countplot	Frequências simples
boxplot	Quartis e outliers
violinplot	Ver forma da distribuição
stripplot	Pontos individuais
swarmplot	Pontos sem sobreposição
barplot	Comparar médias com IC

Recomendações:

- **Rápido:** countplot, boxplot
- **Completo:** violinplot
- **Detalhado:** violin + swarm
- **Estatístico:** barplot (com IC)

Relational Plots

Visualizar relações entre variáveis numéricas:

Função	O que faz	Similar a
scatterplot	Scatter avançado	Matplotlib scatter
lineplot	Linha com IC	Matplotlib plot
relplot	Versátil (scatter/line)	—

Vantagens sobre Matplotlib:

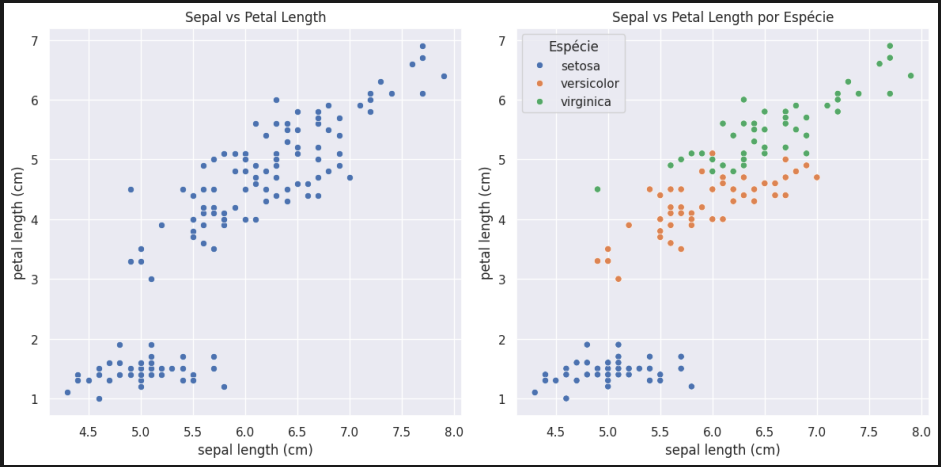
- Fácil adicionar dimensões (hue, size, style)
- Intervalos de confiança automáticos
- Integração perfeita com DataFrames

scatterplot: Scatter Avançado

Python

```python
# Scatter simples
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df,
                x='sepal length (cm)',
                y='petal length (cm)')
plt.title('Sepal vs Petal Length')
plt.show()
# Com cor por categoria (hue)
sns.scatterplot(data=df,
                x='sepal length (cm)',
                y='petal length (cm)',
                hue='species',     # Cor por especie
                palette='deep')
plt.title('Sepal vs Petal Length por Especie')
plt.show()
```

scatterplot: Scatter Avançado (cont.)

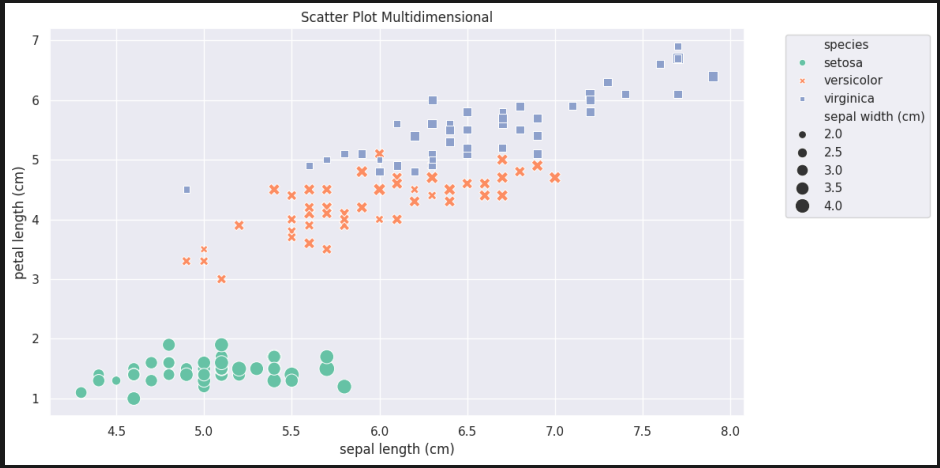


Dimensões Adicionais: Hue, Size, Style

</> Python

```python
# Adicionar múltiplas dimensões - 5 dimensões em um gráfico!
plt.figure(figsize=(12, 6))
sns.scatterplot(data=df,
                x='sepal length (cm)',
                y='petal length (cm)',
                hue='species',          # Cor por especie
                size='sepal width (cm)',  # Tamanho por largura
                style='species',        # Forma por especie
                palette='Set2',
                sizes=(50, 200))        # Range de tamanhos

plt.title('Scatter Plot Multidimensional')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

scatterplot: Scatter Avançado (cont.)



lineplot: Linhas com Intervalo de Confiança

Python

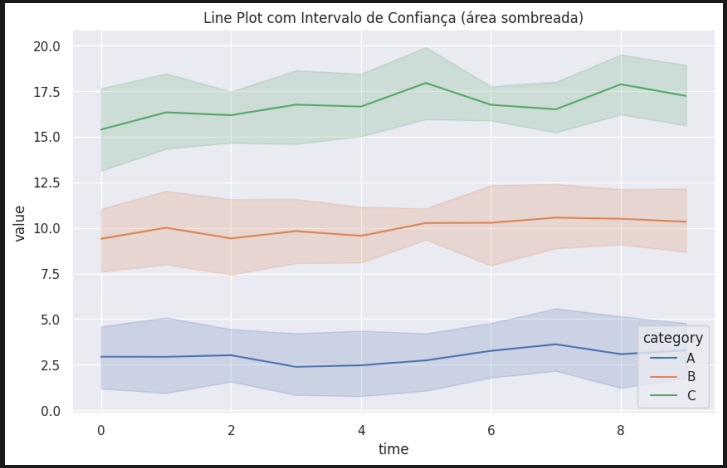
```python
np.random.seed(42)
time_data = pd.DataFrame({
    'time': np.tile(np.arange(10), 3 * 5),          # 150 pontos
    'category': np.repeat(['A', 'B', 'C'], 50),     # 3 grupos
})

# Tendências + ruído acumulado
trend = np.repeat(np.linspace(0, 20, 50), 3)        # mesma forma (150,)
time_data['value'] = trend + np.random.randn(150)
```

lineplot: Linhas com Intervalo de Confiança (cont.)

Python

```python
# Gráfico com área de incerteza
plt.figure(figsize=(10, 6))
sns.lineplot(
    data=time_data,
    x='time',
    y='value',
    hue='category',
    errorbar=('ci', 95)
)
plt.title('Line Plot com Intervalo de Confiança (área sombreada)')
plt.show()
```

lineplot: Linhas com Intervalo de Confiança (cont.)

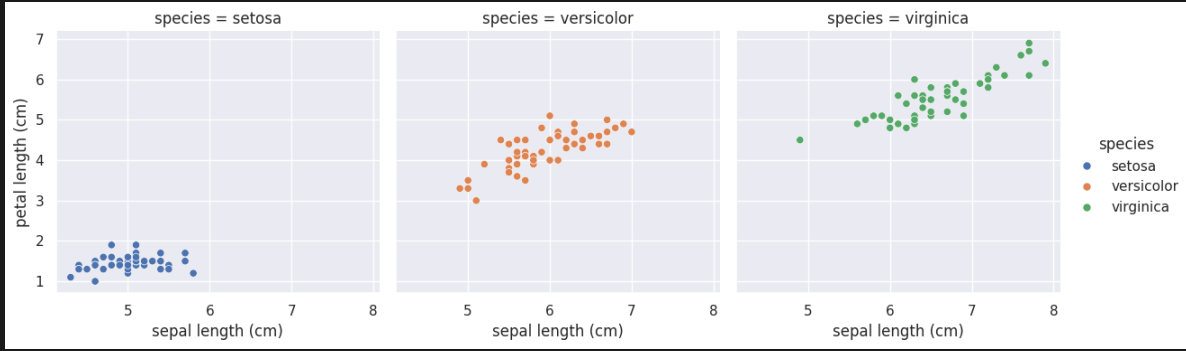


relplot: Scatter ou Line Versátil

Python

```python
# relplot pode fazer scatter ou line
# Vantagem: facil criar facetas (subplots)
# Scatter com facetas
sns.relplot(data=df,
            x='sepal length (cm)',
            y='petal length (cm)',
            hue='species',
            col='species',      # Uma coluna por especie
            kind='scatter',     # Tipo: scatter
            height=4)
plt.show()
# Cada especie em seu proprio subplot
# Util para ver detalhes de cada grupo
```

relplot: Scatter ou Line Versátil (cont.)



Exercício: Relational Plots

Exercício Prático

Explore relações:

1. Crie scatterplot de `petal length` vs `petal width`
2. Adicione `hue='species'`
3. Adicione `size='sepal length (cm)'`
4. Crie relplot com `col='species'`
5. Responda:
   - A correlação é visível?
   - Adicionar size ajudou a ver padrões?
   - Separar por espécie revelou algo novo?

Dica: Múltiplas dimensões revelam padrões ocultos

Resumo: Relational Plots

Principais funções:

- **scatterplot():** Scatter com múltiplas dimensões
- **lineplot():** Linhas com IC automático
- **relplot():** Versátil, cria facetas facilmente

Parâmetros poderosos:

- **hue:** Colorir por categoria
- **size:** Tamanho por variável
- **style:** Forma por categoria
- **col/row:** Criar facetas (subplots)

Quando usar:

- Ver correlações
- Adicionar dimensões extras
- Comparar grupos

Heatmap: Visualizar Matrizes

Heatmap = matriz colorida

O que é:

- Cada célula da matriz vira um quadrado colorido
- Cor indica valor (quente = alto, frio = baixo)
- Perfeito para matriz de correlação

Vantagens:

- Ver padrões rapidamente
- Identificar valores altos/baixos visualmente
- Comparar múltiplas variáveis de uma vez

Quando usar:

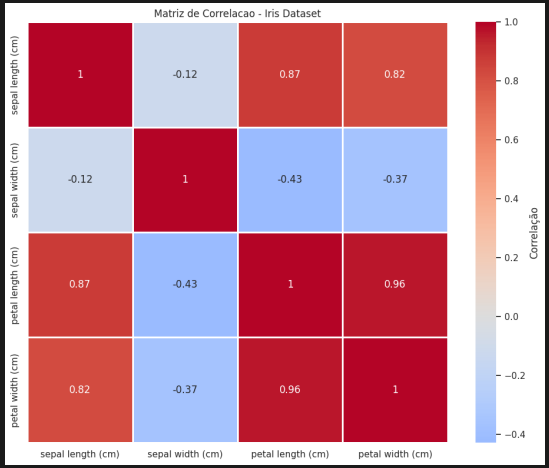
- Matriz de correlação
- Comparar múltiplos grupos
- Dados tabulares com muitas variáveis

Criar Heatmap da Matriz de Correlação

Python

```python
# Calcular matriz de correlacao
numeric_cols = ['sepal length (cm)', 'sepal width (cm)',
                'petal length (cm)', 'petal width (cm)']
corr_matrix = df[numeric_cols].corr()
# Criar heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix,
            annot=True,         # Mostrar valores
            cmap='coolwarm',    # Paleta: azul-branco-vermelho
            center=0,           # Branco = 0
            square=True,        # Celulas quadradas
            linewidths=1,       # Linhas entre celulas
            cbar_kws={'label': 'Correlação'})
plt.title('Matriz de Correlacao - Iris Dataset')
plt.tight_layout()
plt.show()
```

Criar Heatmap da Matriz de Correlação (cont.)



Interpretar Cores do Heatmap

Paleta coolwarm (azul-branco-vermelho):

Como ler:

- **Vermelho escuro:** Correlação positiva forte (↑↑)
- **Azul escuro:** Correlação negativa forte (↑↓)
- **Branco/Claro:** Sem correlação (0)
- **Diagonal:** Sempre vermelho (variável consigo mesma = 1.0)

Dica: Procure por blocos vermelhos fora da diagonal!

Customizar Heatmap

Python

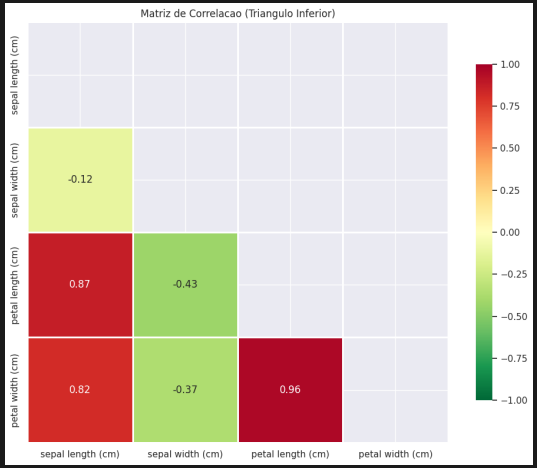
```python
# Heatmap customizado
plt.figure(figsize=(10, 8))

# Mascara para triangulo superior (evitar duplicacao)
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

sns.heatmap(corr_matrix,
            mask=mask,           # Mostrar so triangulo inferior
            annot=True,
            fmt='.2f',           # 2 casas decimais
            cmap='RdYlGn_r',     # Verde-Amarelo-Vermelho
            vmin=-1, vmax=1,     # Escala fixa
            square=True,
            linewidths=2,
            cbar_kws={'shrink': 0.8})
```

Customizar Heatmap (cont.)

Python

```python
plt.title('Matriz de Correlacao (Triangulo Inferior)')
plt.tight_layout()
plt.show()
```

Customizar Heatmap (cont.)



Exercício: Heatmap

Exercício Prático

Crie e interprete heatmap:

1. Calcule matriz de correlação do Iris
2. Crie heatmap com anotações
3. Use `cmap='coolwarm'`
4. Responda:
   - Qual par tem correlação mais forte?
   - Há correlações negativas?
   - Alguma correlação surpreendente?
   - Quais variáveis são mais independentes?
5. Tente com `mask` para triangulo inferior

Dica: Cores intensas = correlações fortes

Bloco 3

Análise Multivariada Simplificada

O que é Análise Multivariada?

Multivariada = múltiplas variáveis juntas

Progressão:

1. **Univariada:** Uma variável (histograma)
2. **Bivariada:** Duas variáveis (scatter)
3. **Multivariada:** Três ou mais variáveis

Por que fazer?

- Ver o ”quadro completo”
- Identificar padrões complexos
- Entender interações entre variáveis
- Preparar para modelagem

Ferramentas principais:

- **Pairplot:** Ver tudo de uma vez
- **FacetGrid:** Análise condicional
- **Dashboards:** Múltiplos gráficos organizados

Pairplot: Ver Tudo de Uma Vez

O que é pairplot?

- Matriz de scatter plots
- Todas as combinações de variáveis
- Diagonal = distribuições
- Off-diagonal = scatter plots

Exemplo - Iris com 4 variáveis:

Grade 4×4 = 16 gráficos em uma figura

Vantagens:

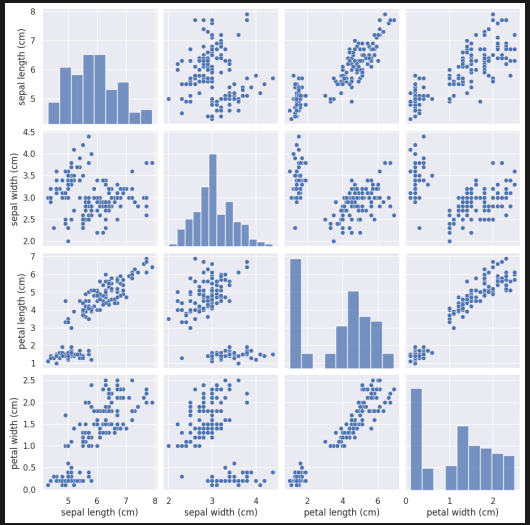
- Ver todas as relações rapidamente
- Identificar variáveis correlacionadas
- Comparar distribuições
- 1 linha de código!

Criar Pairplot do Iris

Python

```python
# Pairplot simples
sns.pairplot(df)
plt.show()
```

Criar Pairplot do Iris (cont.)

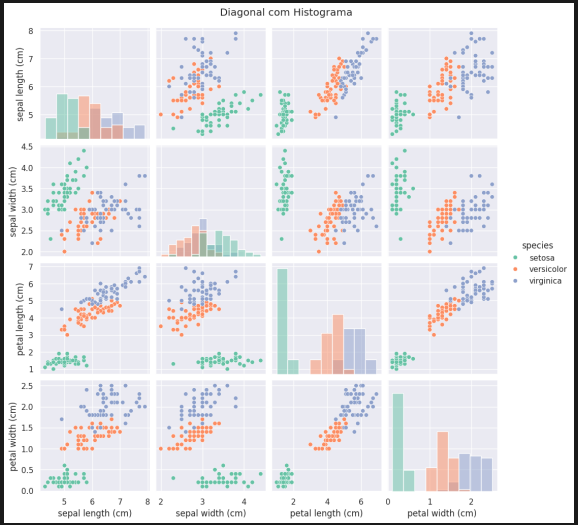


Criar Pairplot do Iris (cont.)

Python

```python
# Com cores por especie
# Histograma na diagonal
sns.pairplot(df, hue='species', diag_kind='hist', palette='Set2')
plt.suptitle('Diagonal com Histograma', y=1.02)
plt.show()
```

Criar Pairplot do Iris (cont.)

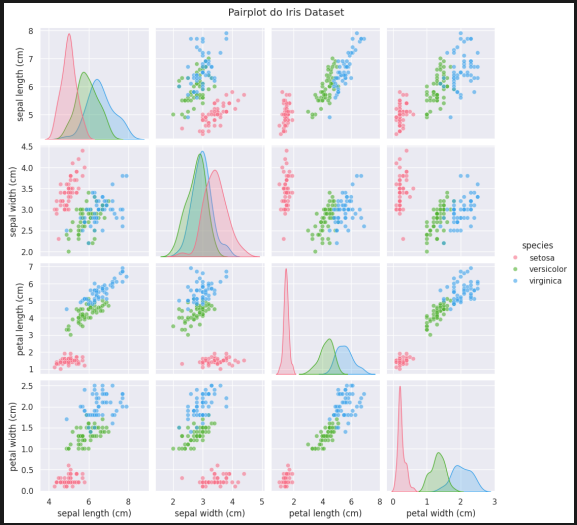


Criar Pairplot do Iris (cont.)

Python

```python
# Customizado
sns.pairplot(df,
             hue='species',
             palette='husl',
             diag_kind='kde',      # KDE na diagonal
             plot_kws={'alpha': 0.6},  # Transparencia
             height=2.5)           # Tamanho de cada subplot
plt.suptitle('Pairplot do Iris Dataset', y=1.02)
plt.show()
```

Criar Pairplot do Iris (cont.)



Interpretar Pairplot

O que procurar:

1. **Diagonal (distribuições):**
   - Forma de cada variável
   - Qual espécie se separa melhor
   - Overlap entre grupos
2. **Off-diagonal (scatter plots):**
   - Correlações fortes (pontos em linha)
   - Separação de grupos (cores separadas)
   - Outliers (pontos isolados)
   - Padrões não-lineares
3. **Padrão geral:**
   - Quais variáveis melhor discriminam grupos?
   - Há variáveis redundantes (altamente correlacionadas)?
   - Dados são separáveis?

Insights do Pairplot do Iris

O que pairplot revela sobre Iris:

1. Separabilidade:

- **Setosa:** Completamente separada (fácil!)
- **Versicolor vs Virginica:** Algum overlap (difícil)

2. Melhores variáveis:

- **Petal length e petal width:** Excelente separação
- **Sepal width:** Pouca discriminação

3. Correlações:

- Petal length $\leftrightarrow$ Petal width: muito forte
- Sepal length $\leftrightarrow$ Petal length: forte
- Pode usar menos variáveis (redundância)

Decisão para ML: Usar petal measurements é suficiente!

FacetGrid: Conceito Básico

FacetGrid = subplots organizados por categoria

Ideia:

- Dividir dados por categoria
- Um subplot para cada grupo
- Mesmo tipo de gráfico em cada
- Fácil comparação visual

Exemplo:

3 espécies $\rightarrow$ 3 subplots lado a lado
Cada um mostra distribuição de petal length

Quando usar:

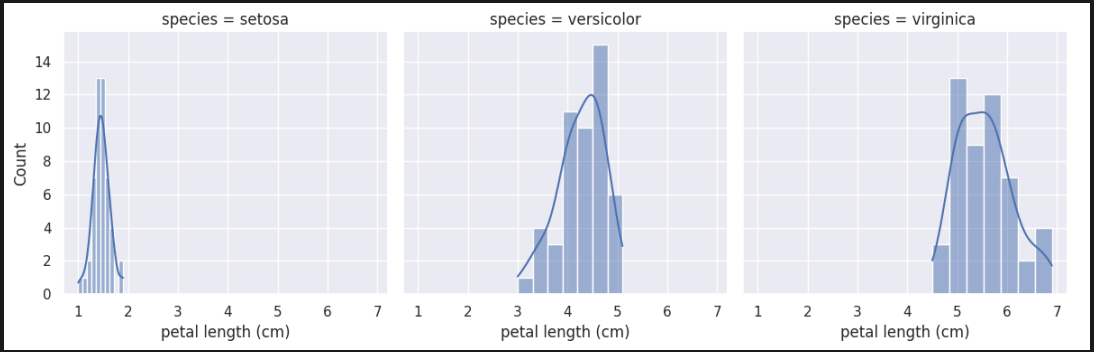
- Comparar grupos
- Ver como relação muda por categoria
- Análise condicional (”para cada X, como é Y?”)

Criar FacetGrid Simples

Python

```python
# FacetGrid com histogramas
g = sns.FacetGrid(df, col='species', height=4, aspect=1)
g.map(sns.histplot, 'petal length (cm)', kde=True)
g.add_legend()
plt.show()
```

Criar FacetGrid Simples (cont.)

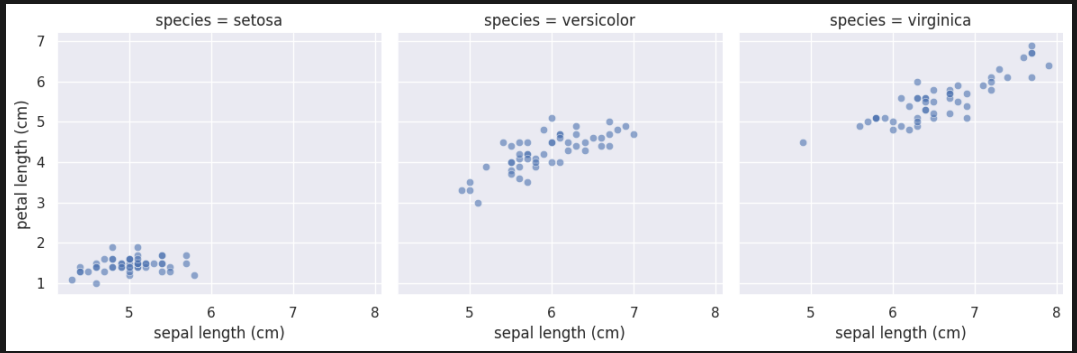


Criar FacetGrid Simples (cont.)

Python

```python
# FacetGrid com scatter
g = sns.FacetGrid(df, col='species', height=4)
g.map(sns.scatterplot, 'sepal length (cm)',
      'petal length (cm)', alpha=0.6)
plt.show()
```

Criar FacetGrid Simples (cont.)



Dashboard Simples com Subplots

Python

```python
# Criar dashboard 2x2
fig, axes = plt.subplots(2, 2, figsize=(14, 10))

# 1. Distribuicoes
sns.histplot(data=df, x='petal length (cm)',
             hue='species', ax=axes[0,0], kde=True)
axes[0,0].set_title('Distribuicao de Petal Length')

# 2. Boxplots
sns.boxplot(data=df, x='species', y='petal width (cm)',
            ax=axes[0,1], palette='Set2')
axes[0,1].set_title('Petal Width por Especie')
```

Dashboard Simples com Subplots (cont.)

Python

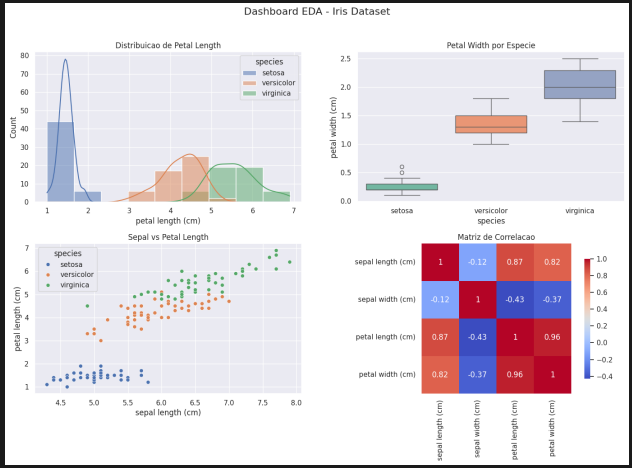
```python
# 3. Scatter
sns.scatterplot(data=df, x='sepal length (cm)',
                y='petal length (cm)', hue='species',
                ax=axes[1,0], palette='deep')
axes[1,0].set_title('Sepal vs Petal Length')
```

Dashboard Simples com Subplots (cont.)

Python

```python
# 4. Heatmap de correlacao
corr_matrix = df[numeric_cols].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm',
            ax=axes[1,1], square=True, cbar_kws={'shrink': 0.8})
axes[1,1].set_title('Matriz de Correlacao')

plt.suptitle('Dashboard EDA - Iris Dataset',
             fontsize=16, y=1.02)
plt.tight_layout()
plt.show()

# 4 perspectivas diferentes em uma figura!
# Distribuicao + Comparacao + Correlacao + Relacoes
```

Dashboard Simples com Subplots (cont.)



Caso de Uso: Relatório Executivo

Cenário: Apresentar análise de vendas para CEO

Dashboard com 4 gráficos:

1. **Top esquerdo:** Vendas por região (bar chart)
   - Mostra onde estamos mais fortes
2. **Top direito:** Tendência temporal (line plot)
   - Crescimento ou queda?
3. **Bottom esquerdo:** Distribuição de ticket médio (hist + kde)
   - Quanto clientes gastam tipicamente
4. **Bottom direito:** Relação visitas × gastos (scatter)
   - Identificar segmentos de clientes

Resultado: CEO vê 4 insights em 30 segundos!

Exercício: Dashboard do Iris

Exercício Prático

Crie dashboard completo:

1. Crie pairplot colorido por espécie
2. Interprete:
   - Qual par de variáveis tem melhor separação?
   - Alguma variável é redundante?
3. Crie dashboard 2×2 com:
   - Histplot de sua escolha
   - Violinplot comparando espécies
   - Scatter plot com hue
   - Heatmap de correlação
4. Documente 3 insights principais

Resumo: Análise Multivariada

Ferramentas principais:

- **Pairplot:** Ver todas as relações (scatter matrix)
- **FacetGrid:** Subplots por categoria
- **Dashboards:** Múltiplos gráficos organizados

Quando usar:

- Exploração inicial completa (pairplot)
- Comparar grupos (FacetGrid)
- Apresentações (dashboard)
- Relatórios executivos

Benefícios:

- Ver múltiplas perspectivas
- Identificar padrões complexos
- Comunicar insights rapidamente

Bloco 4

Comunicando Insights

Storytelling com Dados

EDA não é só fazer gráficos - é contar histórias!

Elementos de uma boa história com dados:

1. **Contexto:** Por que estamos analisando?
2. **Pergunta:** O que queremos saber?
3. **Exploração:** O que os dados mostram?
4. **Insights:** O que descobrimos?
5. **Ação:** O que fazer com essa informação?

Regra de ouro:

Dados contam fatos, você conta a história

Escolher os Gráficos Certos

Guia rápido de decisão:

Objetivo	Gráfico
Mostrar distribuição	Histograma, KDE, Violin
Comparar grupos	Boxplot, Violin, Bar
Mostrar relação	Scatter, Heatmap
Mostrar tendência temporal	Line plot
Mostrar composição	Stacked bar, Pie (evite)
Ver tudo junto	Pairplot, Dashboard
Identificar outliers	Boxplot, Scatter
Comparar frequências	Bar chart, Count plot

Dica: Escolha o gráfico que responde SUA pergunta!

Exemplo: Análise Completa Narrada

Pergunta: ”Podemos classificar espécies de Iris pelas medidas?”

Análise passo a passo:

1. **Exploração inicial:**
   - 150 flores, 50 de cada espécie (balanceado ✓)
   - 4 medidas, sem valores faltantes (limpo ✓)
2. **Distribuições (histograms):**
   - Petal measurements variam mais que sepal
   - Setosa claramente menor em pétalas
3. **Correlações (heatmap):**
   - Petal length ↔ petal width: 0.96 (muito forte!)
   - Sepal width pouco correlacionada
4. **Separação (pairplot):**
   - Setosa: totalmente separável
   - Versicolor × Virginica: algum overlap

Exemplo: Insights e Conclusões

Continuando a análise:

5. Insights principais:

- ✓ Classificação é viável
- ✓ Petal measurements são mais discriminativas
- ✓ Setosa é trivial de classificar
- ⚠Versicolor e Virginica precisam ambas as medidas

6. Recomendação:

- Usar todas as 4 features
- Esperar 95% de acurácia (Setosa 100%, outras 90%)
- Petal width + petal length já dão bom resultado

7. Próximos passos:

- Treinar modelo de classificação
- Validar em dados novos

Do Exploratório ao Relatório

Transformar EDA em relatório:

O que NÃO incluir:

- ✗ Todos os gráficos que você fez
- ✗ Tentativas e erros
- ✗ Código (a menos que seja tutorial)
- ✗ Gráficos sem conclusão

O que INCLUIR:

- ✓ Resumo executivo (1 parágrafo)
- ✓ 3-5 gráficos mais importantes
- ✓ Insights claros para cada gráfico
- ✓ Recomendações acionáveis
- ✓ Próximos passos

Do Exploratório ao Relatório

⚠ Atenção

Menos é mais! Foque no que importa para a decisão.

Boas Práticas de Comunicação

Para visualizações:

1. **Títulos claros:** ”O que este gráfico mostra”
2. **Labels nos eixos:** Sempre com unidades
3. **Legendas:** Quando há múltiplas séries
4. **Cores consistentes:** Mesma cor = mesmo grupo
5. **Fonte legível:** Tamanho adequado

Para texto:

1. **Bullet points:** Mais fácil de escanear
2. **Negrito:** Para destacar números-chave
3. **Contexto primeiro:** Explique antes de mostrar
4. **Insight depois:** Diga o que o gráfico significa
5. **Ação:** O que fazer com essa informação

Erros Comuns em Visualização

O que EVITAR:

1. **Gráficos 3D:** Difícil ler, raramente útil
2. **Pie charts com muitas fatias:** Use bar chart
3. **Eixo Y truncado enganoso:** Sempre comece do zero em bar charts
4. **Cores não distinguíveis:** Use paletas adequadas
5. **Muito texto:** Visualização deve ser visual!
6. **Sem título/labels:** Sempre identifique
7. **Chartjunk:** Efeitos desnecessários
8. **Escalas diferentes:** Dificulta comparação

⚠ Atenção

Visualização ruim é pior que tabela! Seja claro ou não visualize.

Checklist de EDA Completa

Antes de considerar EDA "completo":

1. ☐ Entendi dimensões e tipos dos dados
2. ☐ Identifiquei e tratei valores faltantes
3. ☐ Calculei estatísticas descritivas
4. ☐ Visualizei distribuições de cada variável
5. ☐ Identifiquei e investiguei outliers
6. ☐ Analisei correlações entre variáveis
7. ☐ Comparei grupos (se aplicável)
8. ☐ Criei visualizações multivariadas
9. ☐ Documentei insights principais
10. ☐ Propus próximos passos

Nota Importante

Use este checklist para suas entregas!

Resumo: Aulas 07 e 08

Jornada completa de EDA:

Aula 07:

- Estatística descritiva (média, mediana, quartis)
- Visualizações básicas (histograma, boxplot, bar)
- Matplotlib fundamentals

Aula 08:

- Correlação e relações entre variáveis
- Seaborn (distribution, categorical, relational, matrix)
- Análise multivariada (pairplot, dashboard)
- Comunicação de insights

Agora você pode:

- Explorar qualquer dataset com confiança
- Criar visualizações profissionais
- Contar histórias com dados

Próxima Aula: Pré-processamento

Aula 09 (próxima semana):

- Limpeza de dados
- Tratamento de valores faltantes (imputation)
- Encoding de variáveis categóricas
- Normalização e padronização
- Feature engineering avançado
- Pipeline de pré-processamento

Depois de EDA... preparar dados para ML!

A Seguir: Aula Prática

✋ Exercício Prático

Tempo: 60 minutos
Entrega: via Moodle (notebook)

Tarefas:

1. (atualizado durante a aula)

Notebook: Disponível no Moodle

Obrigado!

Próxima aula: Pré-processamento de Dados