Programação para Ciência de Dados

Análise Exploratória de Dados (EDA) - Parte 1

Arthur Casals

28 de Outubro de 2025

Avisos

- Segunda entrega quinzenal: 31/10 (sexta-feira)
- Última hora de hoje: prática com Iris Dataset
- Dataset de hoje: Iris Dataset
- Próxima aula (30/10): EDA Parte 2 (Seaborn)
- Foco hoje: conceitos fundamentais + muita prática

Agenda

- Introdução: O que é EDA?
- Conhecendo Seus Dados
- Estatística Descritiva Fundamental
- Visualização Básica com Matplotlib

O que é Análise Exploratória de Dados?

EDA = Exploratory Data Analysis

É o processo de **investigar** seus dados para:

- Entender o que você tem
- Descobrir padrões escondidos
- Identificar problemas
- Fazer perguntas certas
- Decidir próximos passos

⚠ Atenção

EDA é como ser um detetive: você procura pistas nos dados!

Por que EDA é Crítica?

Imagine construir uma casa sem conhecer o terreno...

Sem EDA, você pode:

- Criar modelos com dados ruins
- Perder insights importantes
- Tomar decisões erradas
- Desperdiçar tempo com problemas evitáveis
- Não entender por que seu modelo falha

Com EDA, você:

- Conhece a qualidade dos seus dados
- Identifica variáveis importantes
- Detecta outliers e anomalias
- Cria melhores modelos
- Conta histórias com dados

Caso Motivador: Titanic

Como análise de dados salvou vidas?

Após o desastre do Titanic em 1912, análises dos dados revelaram:

- **Mulheres e crianças primeiro** funcionou
  - 74% das mulheres sobreviveram
  - 19% dos homens sobreviveram
- **Classe social importou muito**
  - 63% da 1ª classe sobreviveram
  - 24% da 3ª classe sobreviveram
- **Localização no navio**
  - Quem embarcou em Cherbourg teve mais sorte

Resultado: Novas regulamentações de segurança marítima!

Workflow de um Projeto de Data Science

1. Entender o Problema
O que queremos resolver?
↓
2. Coletar Dados
De onde vêm os dados?
↓
3. EDA (Você está aqui!)
Explorar e entender os dados

Workflow de um Projeto de Data Science

3. EDA (Você está aqui!)
Explorar e entender os dados

↓

4. Pré-processar
Limpar e preparar

↓

5. Modelar
Machine Learning

↓

6. Comunicar
Apresentar resultados

Perguntas que EDA Responde

Durante EDA, você vai perguntar:

1. **Quais dados eu tenho?**
   - Quantas linhas? Quantas colunas?
   - Que tipo de variáveis?
2. **Os dados são confiáveis?**
   - Há valores faltantes?
   - Há erros óbvios?
3. **Qual é a distribuição?**
   - Valores típicos? Extremos?
4. **Há relações interessantes?**
   - Variáveis correlacionadas?
5. **Há algo estranho?**
   - Outliers? Padrões inesperados?

Tipos de Análise

Três níveis de análise:

1. Análise Univariada
- Uma variável por vez
- ”Como é a idade dos passageiros?”
- Ferramentas: histogramas, média, mediana

2. Análise Bivariada
- Relação entre duas variáveis
- ”Idade afeta sobrevivência?”
- Ferramentas: scatter plots, correlação

3. Análise Multivariada
- Múltiplas variáveis juntas
- ”Como idade, sexo e classe afetam sobrevivência?”
- Ferramentas: gráficos complexos, modelos

Ferramentas de Hoje

Python + Pandas + Matplotlib

Pandas:

- Carregar dados
- Calcular estatísticas
- Filtrar e agrupar
- Preparar para visualização

Matplotlib:

- Criar gráficos
- Visualizar distribuições
- Identificar padrões
- Comunicar resultados

Nota Importante

Próxima aula: Seaborn (visualizações ainda mais bonitas!)

Dataset de Hoje: Iris

O dataset mais famoso do mundo!

História:

- Criado por Ronald Fisher em 1936
- 150 flores Iris de 3 espécies diferentes
- Usado para ensinar estatística há 90 anos

Dados:

- **50 flores** de cada espécie:
  - Iris Setosa
  - Iris Versicolor
  - Iris Virginica
- **4 medidas** em centímetros:
  - Comprimento da sépala
  - Largura da sépala
  - Comprimento da pétala
  - Largura da pétala

Por que Iris é Perfeito para Aprender?

Características que facilitam o aprendizado:

- ✓ **Pequeno:** 150 linhas (cabe na tela!)
- ✓ **Limpo:** Sem valores faltantes
- ✓ **Balanceado:** 50 de cada espécie
- ✓ **Simples:** Fácil de entender
- ✓ **Interessante:** Padrões claros
- ✓ **Real:** Dados reais de botanistas

⚠ Atenção

Depois de dominar Iris, você estará pronto para datasets complexos como Wine Quality!

Objetivos da Aula

Ao final desta aula, você será capaz de:

1. **Explorar** qualquer dataset com confiança
2. **Calcular e interpretar** estatísticas básicas:
   - Média, mediana, moda
   - Desvio padrão
   - Quartis
3. **Identificar** valores faltantes e outliers
4. **Criar** visualizações básicas:
   - Histogramas
   - Boxplots
   - Bar charts
5. **Comunicar** insights encontrados

Bloco 1

Conhecendo Seus Dados

Primeiro Passo: Carregar os Dados

Sempre comece carregando seus dados!

Fontes comuns:

- Arquivos CSV (mais comum)
- Excel (.xlsx, .xls)
- Bancos de dados (SQL)
- APIs
- Bibliotecas Python (sklearn, seaborn)

Hoje: Vamos usar o Iris que vem com scikit-learn

Carregar Iris Dataset

Python

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# Carregar Iris do sklearn
from sklearn.datasets import load_iris
# Carregar dados
iris = load_iris()
# Criar DataFrame
df = pd.DataFrame(
    iris.data,
    columns=iris.feature_names
)
# Adicionar coluna de especies
df['species'] = iris.target_names[iris.target]
print("Dados carregados com sucesso!")
```

Primeira Olhada: head()

Ver as primeiras linhas:

</> Python

```python
# Primeiras 5 linhas (padrao)
print(df.head())

# Primeiras 10 linhas
print(df.head(10))

# Por que head() e importante?
# - Ver estrutura dos dados
# - Confirmar que carregou corretamente
# - Ver nomes das colunas
# - Ver tipos de valores
```

Primeira Olhada: head()

Nota Importante

Dica: Sempre execute head() logo após carregar dados!

Primeira Olhada: tail() e sample()

</> Python

```python
# Ultimas 5 linhas
print(df.tail())

# Ultimas 3 linhas
print(df.tail(3))

# Linhas aleatorias (muito util!)
print(df.sample(5))

# Por que sample() é util?
# - Ver dados do meio do dataset
# - Evitar viés das primeiras linhas
# - Verificar variedade dos dados
```

Entender a Estrutura: shape

Quantas linhas e colunas?

</> Python

```python
# Shape retorna (linhas, colunas)
print(f"Shape: {df.shape}")
# Output: (150, 5)

# Interpretar:
n_linhas = df.shape[0]
n_colunas = df.shape[1]

print(f"Temos {n_linhas} linhas (amostras)")
print(f"Temos {n_colunas} colunas (variaveis)")

# 150 flores, 5 informacoes sobre cada flor
```

Entender a Estrutura: shape

⚠ Atenção

shape é SEMPRE sua segunda linha de código após carregar dados!

Tipos de Variáveis

Existem dois tipos principais:

1. Variáveis Numéricas (Quantitativas)

- Representam **quantidades**
- Pode fazer contas (média, soma, etc.)
- **Exemplos:** idade, preço, temperatura, altura
- **No Iris:** comprimento e largura das pétalas

2. Variáveis Categóricas (Qualitativas)

- Representam **categorias** ou **grupos**
- Não faz sentido fazer contas
- **Exemplos:** cor, sexo, cidade, marca
- **No Iris:** espécie da flor

Tipos de Variáveis: Visualização

Como identificar visualmente:

Numéricas:

5.1
4.9
4.7
4.6
5.0

✓ Números
✓ Pode calcular média
✓ Tem ordem natural

Categóricas:

setosa
setosa
versicolor
virginica
versicolor

✓ Texto/labels
✓ Pode contar frequência
✓ Sem ordem natural

info(): Visão Geral Completa

</> Python

```python
# Info mostra TUDO de uma vez
df.info()
# Output exemplo:
# <class 'pandas.core.frame.DataFrame'>
# RangeIndex: 150 entries, 0 to 149
# Data columns (total 5 columns):
#  #   Column             Non-Null Count  Dtype
# ---  ------             --------------  -----
#  0   sepal length (cm)  150 non-null    float64
#  1   sepal width (cm)   150 non-null    float64
#  2   petal length (cm)  150 non-null    float64
#  3   petal width (cm)   150 non-null    float64
#  4   species            150 non-null    object
# dtypes: float64(4), object(1)
# memory usage: 6.0+ KB
```

Interpretando info()

O que cada parte significa:

- **RangeIndex: 150 entries**
  - 150 linhas no total
- **Data columns (total 5 columns)**
  - 5 colunas (variáveis)
- **Non-Null Count**
  - Quantos valores NÃO estão faltando
  - 150 non-null = nenhum valor faltante!
- **Dtype**
  - float64 = número decimal
  - object = texto (categórica)
- **memory usage**
  - Quanto de memória RAM o DataFrame usa

Valores Faltantes: Por que Acontecem?

Razões comuns para dados faltantes:

- **Erro humano**
  - Esqueceram de preencher
  - Digitação incorreta
- **Problema técnico**
  - Sensor quebrado
  - Falha no sistema
- **Não aplicável**
  - ”Renda do cônjuge”para solteiros
  - ”Data de morte”para vivos
- **Privacidade**
  - Pessoa não quis responder
  - Dados sensíveis removidos

Valores Faltantes: Por que Acontecem?

⚠ Atenção

Valores faltantes podem arruinar sua análise!

Identificar Valores Faltantes

</> Python

```python
# Verificar valores faltantes
print(df.isnull().sum())

# Output:
# sepal length (cm)    0
# sepal width (cm)     0
# petal length (cm)    0
# petal width (cm)     0
# species              0

# Iris nao tem valores faltantes!
# (Por isso é ótimo para aprender)

# Porcentagem de valores faltantes
print(df.isnull().sum() / len(df) * 100)
```

Impacto de Valores Faltantes

O que pode acontecer:

- **Pequena quantidade (< 5%):**
  - Geralmente OK
  - Pode remover ou preencher
- **Quantidade moderada (5-20%):**
  - Cuidado ao remover
  - Melhor preencher estrategicamente
- **Grande quantidade (} 20%):**
  - Problema sério
  - Considere remover a variável inteira
  - Ou coletar mais dados

Exemplo visual:

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓✗ = 10% faltante (OK)
✓ ✓ ✓✗✗ ✓✗ ✓✗✗ = 50% faltante (Problema!)

describe(): Primeira Estatística

</> Python

```python
# Estatisticas descritivas basicas
print(df.describe())

#        sepal length  sepal width  petal length  petal width
# count    150.000000   150.000000    150.000000   150.000000
# mean       5.843333     3.057333      3.758000     1.199333
# std        0.828066     0.435866      1.765298     0.762238
# min        4.300000     2.000000      1.000000     0.100000
# 25%        5.100000     2.800000      1.600000     0.300000
# 50%        5.800000     3.000000      4.350000     1.300000
# 75%        6.400000     3.300000      5.100000     1.800000
# max        7.900000     4.400000      6.900000     2.500000
```

Interpretando describe() - Cada Linha

O que cada estatística significa:

- **count:** Quantos valores não-nulos
  - 150 = todas as flores têm medidas
- **mean:** Média (soma ÷ quantidade)
  - Valor ”típico”ou ”central”
- **std:** Desvio padrão (dispersão)
  - Quanto os valores variam
- **min/max:** Menor e maior valores
  - Range dos dados
- **25%, 50%, 75%:** Quartis
  - Dividem dados em 4 partes iguais
  - 50% = mediana

Exercício Guiado: Exploração Inicial

Live Coding

Vamos praticar juntos!

1. Carregue o Iris dataset
2. Execute head() e interprete
3. Verifique shape
4. Execute info()
5. Verifique valores faltantes
6. Execute describe()
7. Responda:
   - Quantas flores temos?
   - Qual a maior largura de pétala?
   - Há dados faltantes?

Tipos de Dados (dtypes)

Python

```python
# Ver tipos de cada coluna
print(df.dtypes)
# Output:
# sepal length (cm)    float64
# sepal width (cm)     float64
# petal length (cm)    float64
# petal width (cm)     float64
# species              object

# Tipos comuns:
# - int64: numeros inteiros
# - float64: numeros decimais
# - object: texto (geralmente categoricas)
# - bool: True/False
# - datetime64: datas
```

Por que Tipos Importam?

Impacto dos tipos de dados:

- **Operações disponíveis**
  - float/int: pode calcular média
  - object: não pode fazer contas
- **Memória**
  - float64: 8 bytes por valor
  - int32: 4 bytes por valor
- **Performance**
  - Operações em int são mais rápidas
- **Visualizações**
  - Gráficos diferentes para cada tipo

Nota Importante

Se os tipos estiverem errados, converta-os!

Conversão de Tipos Básica

Python

```python
# Exemplo: converter species para categorica
df['species'] = df['species'].astype('category')

# Verificar mudanca
print(df.dtypes)

# Por que categorica e melhor?
# - Usa menos memoria
# - Operacoes mais rapidas
# - Ideal para variaveis com poucos valores unicos

# Exemplo com numericas
# Se tivessemos IDs como float, converter para int:
# df['id'] = df['id'].astype('int')
```

Valores Únicos e Frequências

Python

```python
# Valores unicos em uma coluna
print(df['species'].unique())
# Output: ['setosa' 'versicolor' 'virginica']
# Quantos valores unicos?
print(df['species'].nunique())
# Output: 3
# Frequencia de cada valor
print(df['species'].value_counts())
# Output:
# setosa        50
# versicolor    50
# virginica     50
# Com proporcoes
print(df['species'].value_counts(normalize=True))
```

Resumo Visual do Dataset

Checklist de Exploração Inicial:

- ✓ **head()** - Ver primeiras linhas
- ✓ **shape** - Dimensões (linhas × colunas)
- ✓ **info()** - Tipos e valores não-nulos
- ✓ **isnull().sum()** - Valores faltantes
- ✓ **describe()** - Estatísticas básicas
- ✓ **dtypes** - Tipos de dados
- ✓ **nunique()** - Valores únicos
- ✓ **value_counts()** - Frequências

⚠ Atenção

Execute TODOS esses comandos ao explorar um novo dataset!

Checklist: Conheci Meus Dados?

Perguntas que você deve saber responder:

1. ✓ Quantas linhas (amostras) tenho?
2. ✓ Quantas colunas (variáveis) tenho?
3. ✓ Quais são numéricas? Quais são categóricas?
4. ✓ Há valores faltantes? Quantos?
5. ✓ Qual o range de cada variável numérica?
6. ✓ Quais são as categorias das variáveis categóricas?
7. ✓ Os dados parecem confiáveis?
8. ✓ Há algo estranho ou inesperado?

Nota Importante

Se você pode responder todas, parabéns! Você conhece seus dados.

Bloco 2

Estatística Descritiva - Fundamentos

O que é Estatística Descritiva?

Resumir características dos dados com números

Imagine que você tem 150 números e alguém pergunta: ”Como são esses números?”

Em vez de mostrar todos os 150, você resume com:

- **Onde estão concentrados?** (tendência central)
- **Quão espalhados estão?** (dispersão)
- **Quais são os extremos?** (valores mínimo e máximo)

⚠ Atenção

Estatística descritiva transforma 150 números em 5-6 números informativos!

Tendência Central: O ”Centro”dos Dados

Três maneiras de encontrar o ”centro”:

Média

A ”soma dividida”

`(2+4+6)/3 = 4`

Mediana

O ”valor do meio”

`2, 4, 6`

Moda

O ”mais frequente”

`2,2,3,4`

Nota Importante

Nem sempre são iguais! Cada uma conta uma história diferente.

Média: Conceito Intuitivo

Como calcular a média:

Exemplo simples:

Idades: 10, 15, 20, 25, 30

Soma: $10 + 15 + 20 + 25 + 30 = 100$

Quantidade: 5 pessoas

Média = $100 \div 5 = 20$ anos

Interpretação:

- A idade ”típica”é 20 anos
- Se redistribuíssemos as idades igualmente, cada um teria 20
- É o ”ponto de equilíbrio”

Calcular Média no Pandas

Python

```python
# Media de uma coluna
media_sepal_length = df['sepal length (cm)'].mean()
print(f"Media: {media_sepal_length:.2f} cm")
# Output: Media: 5.84 cm

# Media de todas as colunas numericas
print("\n=== MEDIAS ===")
print(df.mean(numeric_only=True))

# Ou especificar colunas
colunas = ['sepal length (cm)', 'sepal width (cm)',
           'petal length (cm)', 'petal width (cm)']
print(df[colunas].mean())

# Interpretacao: Em media, sépalas tem 5.84 cm
```

Média é Sensível a Valores Extremos!

Exemplo: Salários em uma empresa

Cenário 1 (normal):

Salários: R$ 3.000, 3.500, 4.000, 4.200, 4.500
Média = R$ 3.840
✓ Representa bem o grupo

Cenário 2 (com CEO):

Salários: R$ 3.000, 3.500, 4.000, 4.200, 100.000
Média = R$ 22.940
✗ NÃO representa a maioria!

⚠ Atenção

Um único valor extremo pode distorcer completamente a média!

Mediana: O Valor do Meio

Como encontrar a mediana:

Passo 1: Ordenar os valores

`10, 15, 20, 25, 30`

Passo 2: Pegar o valor do meio

`10, 15, 20, 25, 30`
Mediana = 20

Se tiver número par de valores:

`10, 15, 20, 25, 30, 35`
Média dos dois do meio: $(20 + 25) \div 2 = 22.5$

Interpretação: 50% dos valores estão abaixo, 50% estão acima

Calcular Mediana no Pandas

Python

```python
# Mediana de uma coluna
mediana_sepal_length = df['sepal length (cm)'].median()
print(f"Mediana: {mediana_sepal_length:.2f} cm")
# Output: Mediana: 5.80 cm

# Mediana de todas as colunas numericas
print("\n=== MEDIANAS ===")
print(df[colunas].median())

# Comparar media vs mediana
print(f"\nSepal Length:")
print(f"  Media:   {df['sepal length (cm)'].mean():.2f}")
print(f"  Mediana: {df['sepal length (cm)'].median():.2f}")
# No Iris, sao proximas (dados simetricos)
```

Média vs Mediana: Quando Usar?

Guia de decisão:

Use MÉDIA quando:

- ✓ Dados simétricos
- ✓ Sem outliers
- ✓ Distribuição normal
- ✓ Quer o ”total”

Exemplos:

- Altura de pessoas
- Notas de prova
- Temperatura

Use MEDIANA quando:

- ✓ Dados assimétricos
- ✓ Com outliers
- ✓ Distribuição irregular
- ✓ Quer o ”típico”

Exemplos:

- Salários
- Preços de imóveis
- Renda familiar

Exemplo Lado a Lado

Dados SEM outlier:

`2, 3, 4, 5, 6`

Média: 4.0 | Mediana: 4.0

✓ Praticamente iguais

Dados COM outlier:

`2, 3, 4, 5, 100`

Média: 22.8 | Mediana: 4.0

✗ Muito diferentes!

⚠ Atenção

Se média e mediana são muito diferentes, você provavelmente tem outliers!

Moda: O Valor Mais Comum

Qual aparece mais vezes?

Exemplo 1:

Tamanhos de sapato: `37, 38, 39, 39, 39, 40, 41`
Moda = 39 (aparece 3 vezes)

Exemplo 2:

Cores favoritas: `azul, azul, azul, vermelho, verde`
Moda = azul

Pode ter múltiplas modas:

`2, 2, 2, 3, 4, 4, 4`
Modas = 2 e 4 (bimodal)

Quando a Moda é Útil?

Melhor para dados categóricos:

- **Cores mais vendidas**
  - ”A cor mais popular é azul”
- **Problemas mais frequentes**
  - ”80% dos chamados são sobre senha”
- **Escolha mais comum**
  - ”A maioria escolhe opção A”

Menos útil para dados contínuos:

- Altura: 1.75m, 1.76m, 1.77m... (todos diferentes!)
- Temperatura: valores únicos

Nota Importante

Para categóricas, use value_counts() em vez de mode()

Calcular Moda no Pandas

Python

```python
# Moda (menos util para numericas continuas)
moda = df['sepal length (cm)'].mode()
print(f"Moda: {moda.values}")
# MELHOR: Para categoricas, use value_counts()
print("\n=== ESPECIES (moda categorica) ===")
print(df['species'].value_counts())
# Output:
# setosa        50
# versicolor    50
# virginica     50
# Todas tem mesma frequencia (trimodal)

# Especie mais comum
mais_comum = df['species'].value_counts().index[0]
print(f"\nEspecie mais comum: {mais_comum}")
```

Comparação Visual das Três Medidas

Dados simétricos:

`1, 2, 3, 3, 3, 4, 5`

Moda = 3 | Mediana = 3 | Média = 3

✓ Todas iguais (distribuição simétrica)

Dados assimétricos à direita:

`1, 2, 2, 2, 3, 4, 100`

Moda = 2 | Mediana = 2 | Média = 16.3

Moda < Mediana < Média

Dados assimétricos à esquerda:

`1, 96, 98, 98, 98, 99, 100`

Moda = 98 | Mediana = 98 | Média = 84.3

Média < Mediana < Moda

Caso de Uso: Análise de Preços

Cenário: Análise de preços de apartamentos

Dados:

R$ 200k, 220k, 250k, 280k, 300k, 320k, 5.000k

Análise:

- **Média:** R$ 938k
  - ✗ Muito alto! Não representa maioria
- **Mediana:** R$ 280k
  - ✓ Preço ”típico”do mercado
- **Interpretação:**
  - Um apartamento luxuoso distorce a média
  - Mediana é melhor para reportar

Exercício: Calcular e Comparar Medidas

✋ Exercício Prático

Pratique com o Iris dataset:

1. Calcule média, mediana e moda de `petal length`
2. Compare os três valores
3. São similares ou diferentes?
4. O que isso diz sobre a distribuição?
5. Repita para `sepal width`
6. Qual variável é mais simétrica?

Dica: Use .mean(), .median(), .mode()

Resumo: Tendência Central

Medida	Quando usar	Cuidado
Média	Dados simétricos	Sensível a outliers
Mediana	Dados assimétricos	Ignora extremos
Moda	Dados categóricos	Múltiplas modas

Regra de ouro:

- Sempre calcule MÉDIA E MEDIANA
- Se forem muito diferentes $\rightarrow$ investigate outliers
- Para salários, preços, renda $\rightarrow$ prefira mediana
- Para altura, temperatura, notas $\rightarrow$ média OK

Dispersão: Dados Espalhados ou Concentrados?

Dois grupos com mesma média (50):

Grupo A (concentrado):

`48, 49, 50, 51, 52`

Média = 50 | Todos perto da média

Grupo B (espalhado):

`10, 30, 50, 70, 90`

Média = 50 | Valores muito diferentes

⚠ Atenção

Mesma média, comportamentos MUITO diferentes! Precisamos medir a dispersão.

Visualização: Concentrado vs Espalhado

Dados concentrados (baixa dispersão):

●●●●●●●●

Todos os pontos próximos
Previsível - valores similares

Dados espalhados (alta dispersão):

● ... ● ..... ● .. ●

Pontos distantes entre si
Imprevisível - valores muito variados

Por que importa?

- Baixa dispersão $\rightarrow$ mais confiável, mais homogêneo
- Alta dispersão $\rightarrow$ menos confiável, mais heterogêneo

Range (Amplitude): A Medida Mais Simples

Range = Máximo - Mínimo

Exemplo:

Idades: `18, 22, 25, 30, 45`
Mínimo = 18 | Máximo = 45
Range = $45 - 18 = 27$ anos

Interpretação:

- Os dados variam em 27 anos
- Diferença entre o mais novo e o mais velho

Limitação:

- Usa apenas 2 valores (ignora os outros!)
- Muito sensível a outliers

Limitações do Range

Exemplo mostrando o problema:

Grupo A:

`10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20`

Range = $20 - 10 = 10$

✓ Bem distribuído

Grupo B:

`10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 20`

Range = $20 - 10 = 10$

✗ Quase todos iguais, mas range é o mesmo!

⚠ Atenção

Range NÃO nos diz como os valores estão distribuídos entre **min** e **max**

Variância: Conceito sem Fórmula

Variância mede: Quão longe, em média, os valores estão da média

Ideia intuitiva:

1. Para cada valor, veja quão longe está da média
2. Eleve ao quadrado (para valores negativos não cancelarem positivos)
3. Tire a média desses quadrados

Interpretação:

- **Variância baixa:** valores próximos da média
- **Variância alta:** valores espalhados

Problema:

- Unidade é "ao quadrado" (ex: cm^2)
- Difícil de interpretar
- Solução: Desvio Padrão!

Desvio Padrão: A Dispersão ”Típica”

Desvio Padrão = Raiz quadrada da variância

Por que é melhor?

- Volta para a unidade original (cm, anos, reais)
- Fácil de interpretar
- Mesma escala que a média

Interpretação intuitiva:

”Em média, os valores se desviam X unidades da média”

Exemplo:

- Média de altura = 170 cm
- Desvio padrão = 10 cm
- Interpretação: ”A maioria das pessoas está entre 160 e 180 cm”

Calcular Variância e Desvio Padrão

Python

```python
# Desvio padrao (mais usado)
std_sepal = df['sepal length (cm)'].std()
print(f"Desvio padrao: {std_sepal:.2f} cm")
# Output: Desvio padrao: 0.83 cm

# Variancia (menos interpretavel)
var_sepal = df['sepal length (cm)'].var()
print(f"Variancia: {var_sepal:.2f} cm²")

# Para todas as colunas
print("\n=== DESVIO PADRAO ===")
print(df[colunas].std())

# Interpretação: Sepal length varia ~0.83cm da media
```

Interpretação: Regra 68-95-99.7

Para dados com distribuição normal (em forma de sino):

68% dos dados estão dentro de **1 desvio padrão**
95% dos dados estão dentro de **2 desvios padrão**
99.7% dos dados estão dentro de **3 desvios padrão**

Exemplo prático:

- Altura média = 170 cm, desvio = 10 cm
- 68% das pessoas: entre 160 e 180 cm
- 95% das pessoas: entre 150 e 190 cm
- 99.7% das pessoas: entre 140 e 200 cm

Nota Importante

Valores fora de 3 desvios são considerados muito raros!

Alto Desvio *vs* Baixo Desvio

Comparação visual:

Baixo desvio padrão (dados concentrados):

Média = 50 | DP = 2

`48, 49, 50, 51, 52`

✓ Previsível, homogêneo, confiável

Alto desvio padrão (dados espalhados):

Média = 50 | DP = 30

`10, 30, 50, 70, 90`

✗ Imprevisível, heterogêneo, variável

Quando cada um é bom:

- **Baixo DP:** Controle de qualidade, manufatura
- **Alto DP:** Pode indicar diversidade (às vezes desejável)

Caso de Uso: Controle de Qualidade

Cenário: Fábrica de parafusos (especificação: 10.0 mm)

Máquina A:
- Média: 10.0 mm ✓
- Desvio padrão: 0.1 mm
- Resultado: 9.9, 10.0, 10.1 mm (consistente!)

Máquina B:
- Média: 10.0 mm ✓
- Desvio padrão: 2.0 mm
- Resultado: 8.0, 10.0, 12.0 mm (problema!)

⚠ Atenção

Máquina A é melhor! Mesmo com mesma média, menor dispersão = maior qualidade

Exercício: Comparar Dispersão entre Grupos

✋ Exercício Prático

Compare dispersão por espécie:

1. Calcule média e desvio padrão de `petal length` para cada espécie:

```
for species in df['species'].unique():
    subset = df[df['species'] == species]
    mean = subset['petal length (cm)'].mean()
    std = subset['petal length (cm)'].std()
    print(f"{species}: media={mean:.2f}, std={std:.2f}")

```

2. Qual espécie tem maior dispersão?
3. Qual é mais homogênea?
4. O que isso significa biologicamente?

Resumo: Por que Dispersão Importa

Dispersão nos diz:

- **Confiabilidade:** Baixa dispersão = mais previsível
- **Variabilidade:** Alta dispersão = mais heterogêneo
- **Qualidade:** Em manufatura, menor é melhor
- **Risco:** Em finanças, maior dispersão = maior risco
- **Consistência:** Mede se processo é estável

Lembre-se:

- Range: simples mas limitado
- Variância: difícil de interpretar
- Desvio padrão: **use este!**

Nota Importante

Sempre reporte média E desvio padrão juntos!

Dividir Dados em Partes Iguais

Imagine 100 pessoas em fila por altura:

|···25 pessoas···|···25 pessoas···|···25 pessoas···|···25 pessoas···|

Baixos ← Médios → Altos

Divisões:

- **Q1 (25%):** Marca onde 25% são mais baixos
- **Q2 (50%):** Mediana - metade de cada lado
- **Q3 (75%):** Marca onde 75% são mais baixos

Útil para:

- Entender distribuição dos dados
- Identificar onde você está (ex: ”estou no top 25%”)
- Detectar outliers

Percentis: Conceito Visual

Percentil divide dados em 100 partes

Exemplo - Notas de vestibular:

- **Percentil 90:** Você foi melhor que 90% dos candidatos
- **Percentil 50:** Você está na média (metade acima, metade abaixo)
- **Percentil 10:** 90% foram melhor que você

Interpretação:

"Percentil X significa que X% dos valores estão abaixo"

Aplicações práticas:

- Curvas de crescimento infantil
- Rankings (vestibular, concursos)
- Salários (saber sua posição no mercado)

Quantis: Conceito Geral

Quantil = Valor que divide a distribuição em partes proporcionais

Definição

O quantil de ordem p (com $0 < p < 1$) é o valor x_p tal que:

$$P(X \leq x_p) = p$$

Ou seja, ao menos $p \times 100\%$ dos dados estão abaixo de x_p.

Interpretação

- O quantil indica um ponto de corte da distribuição.
- Exemplos: o quantil de ordem 0,25 corresponde a 25% dos dados abaixo, enquanto o quantil 0,5 é a mediana (50% abaixo e 50% acima).

Quantis: Conceito Geral

Quantil = Valor que divide a distribuição em partes proporcionais

Fórmula geral (amostral)

Dada uma amostra ordenada de n observações, o quantil de ordem p encontra-se na posição:

$$i = p(n + 1)$$

- Se i não for inteiro, interpola-se entre os valores vizinhos.

Exemplo: Para $n = 10$ e $p = 0{,}25$, temos $i = 0{,}25 \times 11 = 2{,}75$, indicando que o 1º quartil está 75% do caminho entre o 2º e o 3º valor.

Tipos Especiais de Quantis

Quantis = divisões iguais dos dados ordenados

Principais tipos de quantis

- **Quartis** — dividem os dados em 4 partes (Q1, Q2, Q3)
- **Quintis** — dividem os dados em 5 partes
- **Decis** — dividem os dados em 10 partes
- **Duo-decis** — dividem os dados em 12 partes
- **Percentis** — dividem os dados em 100 partes

Tipos Especiais de Quantis

Resumo simples

Um **quantil** marca a posição onde uma certa porcentagem dos dados fica abaixo de um valor.

É como traçar “divisões” em uma lista de valores ordenados para entender:

- onde está o centro (mediana),
- quais valores são típicos,
- e onde ficam as extremidades (distribuição dos dados).

Calcular Percentis Customizados

Python

```python
# Percentis customizados
col = 'sepal length (cm)'
percentis = [0.10, 0.25, 0.50, 0.75, 0.90, 0.95, 0.99]
valores = df[col].quantile(percentis)

print("=== PERCENTIS ===")
for p, v in zip(percentis, valores):
    print(f"P{int(p*100):2d}: {v:.2f} cm - "
          f"{int(p*100)}% das flores tem <= {v:.2f}cm")

# Output exemplo:
# P10: 4.80 cm - 10% das flores tem <= 4.80cm
# P25: 5.10 cm - 25% das flores tem <= 5.10cm
# P50: 5.80 cm - 50% das flores tem <= 5.80cm
# ...
```

Quartis: Q1, Q2, Q3

Quartis = Percentis especiais

Q1
Primeiro Quartil
= Percentil 25

25% dos dados
estão abaixo

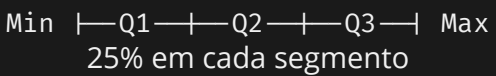
Q2
Segundo Quartil
= Percentil 50

= Mediana
50% de cada lado

Q3
Terceiro Quartil
= Percentil 75

75% dos dados
estão abaixo

Visualização:



Calcular Quartis no Pandas

</> Python

```python
# Quartis de uma coluna
col = 'sepal length (cm)'

q1 = df[col].quantile(0.25)
q2 = df[col].quantile(0.50)  # = mediana
q3 = df[col].quantile(0.75)

print(f"Q1 (25%): {q1:.2f} cm")
print(f"Q2 (50%): {q2:.2f} cm - MEDIANA")
print(f"Q3 (75%): {q3:.2f} cm")

# Interpretacao:
# - 25% das flores tem sepal length <= Q1
# - 50% das flores tem sepal length <= Q2
# - 75% das flores tem sepal length <= Q3
```

Exemplo: Idades de 100 Passageiros

Distribuição dos dados:

- 50 passageiros com **30 anos**
- 50 passageiros com **60 anos**

Ordenando a amostra:

`30, 30, ... , 30 (50x), 60, 60, ... , 60 (50x)`

Posições dos quartis:

- $Q1$: posição 25 $\rightarrow$ **30**
- $Q2$: entre posições 50 e 51 $\rightarrow$ **(30 + 60)/2 = 45**
- $Q3$: posição 75 $\rightarrow$ **60**

Resultado:

$$Q1 = 30, \quad Q2 = 45, \quad Q3 = 60$$

Interpretação: Quartis com Dados Repetidos

Ponto-chave

Os quartis são **valores posicionais**, não contagens exatas de indivíduos. Mesmo com muitos dados iguais, a posição do quartil define o valor.

- $Q1 = 30$: indica que *cerca de 25%* das idades estão abaixo ou iguais a 30.
- $Q2 = 45$: representa a **mediana**, ponto intermediário da amostra.
- $Q3 = 60$: indica que *cerca de 75%* das idades estão abaixo ou iguais a 60.

Mas atenção: Como metade dos passageiros tem exatamente 30 e metade tem 60, o quartil não indica que "25 pessoas têm 30 anos ou menos", e sim que **o ponto de corte do 25º percentil cai em 30**.

Conclusão: Quando há repetições, o quartil pode coincidir com valores que ocorrem muitas vezes. A delimitação é uma posição na ordenação, não um número exato de elementos.

Exemplo: Quartis e Valores Extremos

Cenário: 100 passageiros

- 49 passageiros com **30 anos**
- 49 passageiros com **60 anos**
- 1 passageiro com **1 ano**
- 1 passageiro com **100 anos**

Dados ordenados:

`1, 30, 30, ..., 30, 60, 60, ..., 60, 100`

Quartis obtidos:

$$Q1 = 30, \quad Q2 = 45, \quad Q3 = 60$$

Observação

Mesmo com **um valor extremo muito baixo (1)** e **um muito alto (100)**, os quartis não mudam: continuam refletindo o centro da massa de dados (30 e 60).

Quartis: Robustez a Outliers

Ponto-chave

Os quartis são **robustos**, ou seja, pouco afetados por valores extremos. Isso ocorre porque dependem apenas das posições percentuais (25%, 50%, 75%), e não dos extremos.

- Os outliers alteram o **mínimo**, o **máximo** e o **intervalo total**, mas *não afetam os quartis principais*.
- Essa robustez é útil para detectar anomalias visuais em **boxplots**, pois os outliers aparecem fora dos “bigodes”.

Amplitude Interquartil (IQR)

Definição: a **Amplitude Interquartil (IQR)** mede a dispersão dos 50% centrais dos dados.

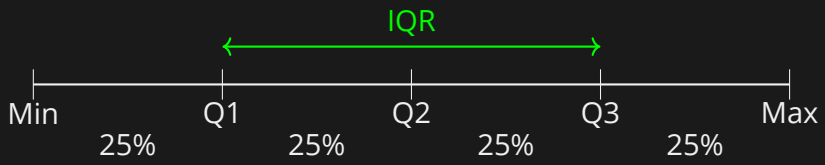
$$IQR = Q3 - Q1$$

Interpretação

- Representa a variação entre o 1º quartil ($Q1$) e o 3º quartil ($Q3$).
- Ignora os 25% menores e os 25% maiores valores.
- É uma medida **robusta**, pouco sensível a outliers (diferente do desvio-padrão).
- Quanto maior o IQR, maior é a variabilidade na parte central da distribuição; quanto menor, mais concentrados estão os dados.

IQR - Visualização

Representação visual dos quartis:



Interpretação:

- Cada segmento contém 25% dos dados
- IQR cobre os 50% do meio
- Min e Max mostram os extremos

IQR - Exemplo

Exemplo: Conjunto de idades dos 100 passageiros

$$Q1 = 30, \quad Q3 = 60 \quad \Rightarrow \quad IQR = 60 - 30 = 30$$

Visualização:

(A faixa central é o IQR e representa os 50% centrais)

IQR - Regra dos 50% Centrais

Por que focar nos 50% centrais?

- **Representam o "típico"**
  - Excluem extremos
  - Foco na maioria dos dados
- **Robustos a outliers**
  - Um valor muito alto não afeta IQR
  - Diferente da média e desvio padrão
- **Fáceis de comunicar**
  - "A maioria está entre X e Y"
  - Mais intuitivo que desvio padrão

Exemplo prático:

"50% dos salários estão entre R$ 3.000 e R$ 5.000"
Muito mais claro que "desvio padrão de R$ 1.500"

Usando o IQR para Detectar Outliers

Método de Tukey (baseado no IQR):

$$\text{Limites aceitáveis} = [Q1 - 1.5 \times IQR,\ Q3 + 1.5 \times IQR]$$

Limites:

- **Limite inferior:** $Q1 - 1.5 \times IQR$
- **Limite superior:** $Q3 + 1.5 \times IQR$

Tudo fora desses limites é considerado outlier

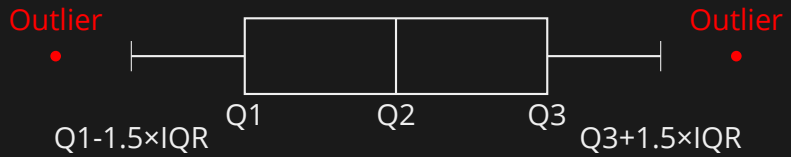
Exemplo:

Q1 = 5.1 | Q3 = 6.4 | IQR = 1.3
Limite inferior = $5.1 - 1.5 \times 1.3 = 3.15$
Limite superior = $6.4 + 1.5 \times 1.3 = 8.35$
Outliers: valores < 3.15 ou > 8.35

Usando o IQR para Detectar Outliers

Por que 1.5?

- Convenção estatística estabelecida
- Captura 99.3% dos dados (se distribuição normal)
- Equilibra sensibilidade: não muito restrito, não muito permissivo



Usando o IQR para Detectar Outliers

No exemplo anterior:

$$Q1 = 30, \quad Q3 = 60, \quad IQR = 30$$

$$\text{Limite inferior} = 30 - 1.5(30) = -15 \quad \Rightarrow \text{mínimo real: 1}$$

$$\text{Limite superior} = 60 + 1.5(30) = 105 \quad \Rightarrow \text{máximo real: 100}$$

Interpretação:

- Valores fora desse intervalo são considerados **outliers**.
- No caso, o passageiro de 1 ano é um *outlier inferior*.
- O passageiro de 100 anos é um *outlier superior*.
- Os quartis e o IQR permanecem inalterados (mostram a **robustez estatística**).

Usando o IQR para Detectar Outliers

Método de Tukey (baseado no IQR):

$$\text{Limites aceitáveis} = [\,Q1 - 1.5 \times IQR,\; Q3 + 1.5 \times IQR\,]$$

No exemplo anterior:

$$Q1 = 30, \quad Q3 = 60, \quad IQR = 30$$

$$\text{Limite inferior} = 30 - 1.5(30) = -15 \quad \Rightarrow \text{mínimo real: } 1$$

$$\text{Limite superior} = 60 + 1.5(30) = 105 \quad \Rightarrow \text{máximo real: } 100$$

Visual:

(Os pontos • representam os outliers detectados)

Aplicação - Quartis e Outliers - Titanic

Python

```python
# Aula 6 - Slide 62

#         count       mean        std   min   q25   q75   max
# Pclass
# 1         186  38.233441  14.802856  0.92  27.0  49.0  80.0
# 2         173  29.877630  14.001077  0.67  23.0  36.0  70.0
# 3         355  25.140620  12.495398  0.42  18.0  32.0  74.0
```

Interpretação dos Resultados

Resumo estatístico - Primeira classe (Pclass = 1):

$$n = 186, \quad \text{média} = 38.23, \quad \text{desvio-padrão} = 14.8$$

$$\text{mínimo} = 0.92, \quad Q1 = 27, \quad Q3 = 49, \quad \text{máximo} = 80$$

1. Interpretação geral dos quartis

- $Q1 = 27 \rightarrow$ 25% dos valores estão abaixo ou iguais a 27.
- $Q3 = 49 \rightarrow$ 75% dos valores estão abaixo ou iguais a 49.
- Metade mais representativa dos dados entre 27 e 49 anos (faixa central de 50% dos valores).

2. Amplitude interquartil (IQR)

$$IQR = Q3 - Q1 = 49 - 27 = 22$$

Essa é a medida da dispersão central (quanto variam os 50% centrais dos dados).

Interpretação dos Resultados (cont.)

Resumo estatístico - Primeira classe (Pclass = 1):

$$n = 186, \quad \text{média} = 38.23, \quad \text{desvio-padrão} = 14.8$$

$$\text{mínimo} = 0.92, \quad Q1 = 27, \quad Q3 = 49, \quad \text{máximo} = 80$$

3. Detecção de outliers (método de Tukey)

$$\text{Limites} = [Q1 - 1.5 \times IQR,\ Q3 + 1.5 \times IQR] = [27 - 33,\ 49 + 33] = [-6,\ 82]$$

Valores abaixo de -6 ou acima de 82 são outliers potenciais.

Interpretação dos Resultados (cont.)

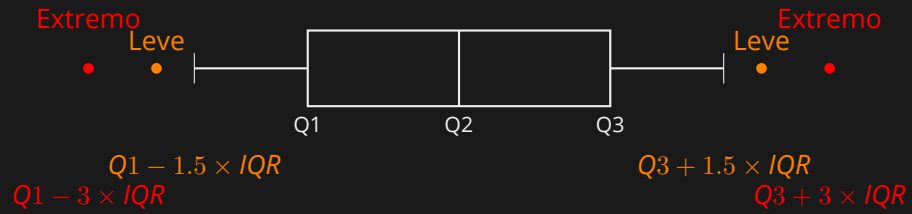
4. Conclusão:

- O intervalo principal dos dados vai aproximadamente de 27 a 49.
- Máximo e mínimo dentro dos limites $\rightarrow$ tecnicamente, não há outliers.
- O valor mínimo (0.92) permanece dentro da faixa do método de Tukey, mas está mais distante do corpo central dos dados (Q1 = 27) do que o máximo (80) está de Q3 = 49.
- A média (38.23) está mais próxima de Q1 (27) do que de Q3 (49), sugerindo uma cauda mais longa à esquerda.
- Desvio-padrão relativamente alto (14.8) comparado ao IQR (22), o que indica dispersão significativa mas concentrada no meio-superior (típico de distribuições assimétricas à esquerda).

Outliers: Leves e Extremos

Como identificar?

- **Outliers leves:** valores situados entre 1.5 e $3 \times$ *IQR* além de $Q1$ ou $Q3$.
- **Outliers extremos:** valores além de $3 \times$ *IQR* para fora de $Q1$ e $Q3$.
- Baseia-se na regra do intervalo interquartil para detectar pontos muito afastados do restante dos dados.



Identificar Outliers no Iris

Python

```python
def detect_outliers_iqr(series):
    Q1 = series.quantile(0.25)
    Q3 = series.quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = series[(series < lower_bound) |
                      (series > upper_bound)]

    return outliers, lower_bound, upper_bound
```

Identificar Outliers no Iris

Python

```python
# Aplicar a sepal width
col = 'sepal width (cm)'
outliers, lower, upper = detect_outliers_iqr(df[col])

print(f"Limites: [{lower:.2f}, {upper:.2f}]")
print(f"Outliers encontrados: {len(outliers)}")
if len(outliers) > 0:
    print(f"Valores: {outliers.values}")
```

Caso de Uso: Detecção de Fraudes

Cenário: Sistema de detecção de transações fraudulentas

Transações normais:

R$ 20, R$ 35, R$ 50, R$ 45, R$ 60, R$ 40
Q1 = R$ 32.5 | Q3 = R$ 52.5 | IQR = R$ 20
Limites: [R$ 2.5, R$ 82.5]

Nova transação: R$ 500

- R$ 500 > R$ 82.5 ✗
- É um outlier!
- Sistema alerta: possível fraude

⚠ Atenção

IQR ajuda a identificar comportamentos anormais automaticamente!

Exercício: Análise de Outliers

✋ Exercício Prático

Detecte outliers em todas as variáveis:

```
colunas = ['sepal length (cm)', 'sepal width (cm)',
           'petal length (cm)', 'petal width (cm)']

for col in colunas:
    outliers, lower, upper = detect_outliers_iqr(df[col])
    print(f"\n{col}:")
    print(f"  Limites: [{lower:.2f}, {upper:.2f}]")
    print(f"  Outliers: {len(outliers)}")

    if len(outliers) > 0:
        print(f"  Valores: {outliers.values}")

```

Exercício: Análise de Outliers (cont.)

Exercício Prático

Detecte outliers em todas as variáveis:
Perguntas:

1. Qual variável tem mais outliers?
2. Os outliers são extremos ou leves?

Resumo: Quartis e Outliers

Conceitos-chave:

- **Quartis (Q1, Q2, Q3):** Dividem dados em 4 partes iguais
- **IQR = Q3 - Q1:** Largura dos 50% centrais
- **Outliers:** Valores fora de $Q1 - 1.5 \times IQR$ e $Q3 + 1.5 \times IQR$

Por que importa?

- Entender distribuição além da média
- Identificar valores anormais
- Comunicar de forma clara (”50% estão entre X e Y”)
- Detectar problemas de qualidade

Ferramentas Pandas:

- `.quantile(0.25)` - Q1
- `.quantile(0.75)` - Q3
- `.describe()` - mostra tudo de uma vez

Bloco 3

Visualização Básica com Matplotlib

Por que Visualizar?

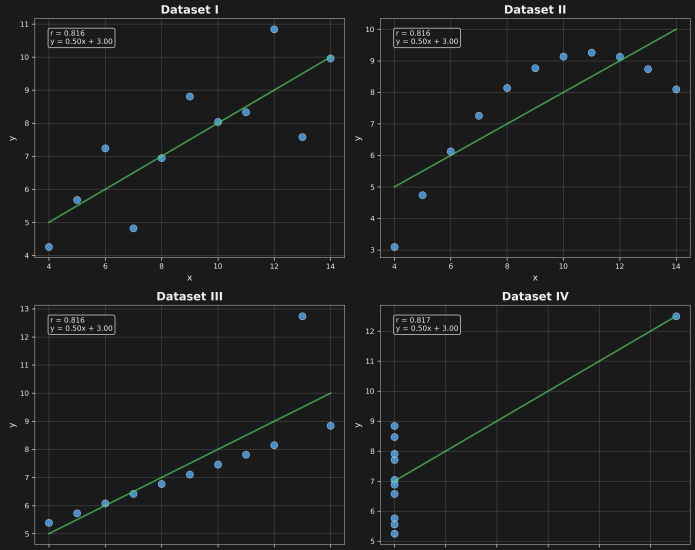
O Quarteto de Anscombe (1973):
Quatro datasets com as **mesmas** estatísticas:

- Mesma média (7.5)
- Mesmo desvio padrão
- Mesma correlação

Mas quando visualizados... completamente diferentes!

- Dataset 1: Linear
- Dataset 2: Curva
- Dataset 3: Linear com outlier
- Dataset 4: Pontos verticais com outlier

O Quarteto de Anscombe



Por que Visualizar?

⚠ Atenção

Números sozinhos não contam a história completa. SEMPRE visualize seus dados!

Matplotlib: A Biblioteca de Visualização

O que é Matplotlib?

- Biblioteca Python para criar gráficos
- A mais usada e estabelecida
- Base para outras bibliotecas (Seaborn, Pandas plotting)
- Controle completo sobre cada elemento

Por que aprender Matplotlib primeiro?

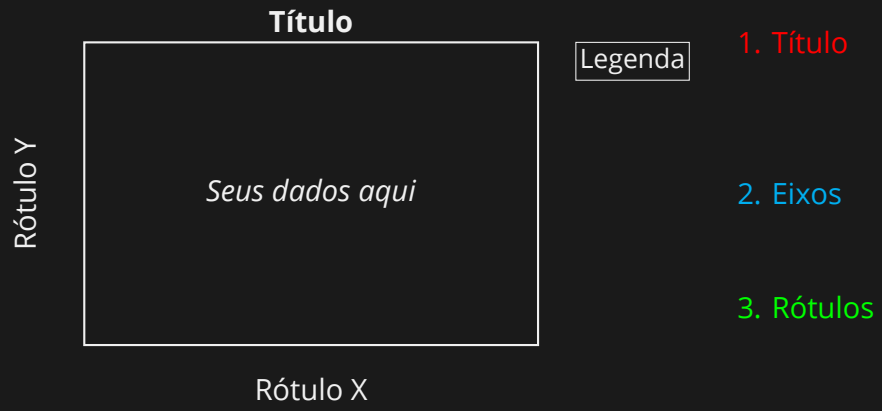
- Fundação para visualização em Python
- Entender os conceitos básicos
- Customização total quando necessário
- Próxima aula: Seaborn (mais fácil, mais bonito)

Nota Importante

Matplotlib = controle total | Seaborn = facilidade

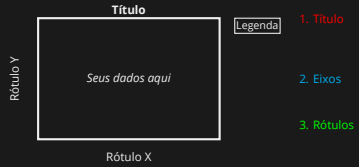
Anatomia de um Gráfico (Simplificado)

Componentes principais:



Anatomia de um Gráfico (Simplificado)

Componentes principais:



Elementos essenciais:

- **Título (Title):** O que o gráfico mostra?
- **Eixos (Axis):** X (horizontal) e Y (vertical)
- **Rótulos (Labels):** O que cada eixo representa?
- **Legenda (Legend):** Identificar séries (quando necessário)

Importar Matplotlib

Python

```python
# Importacao padrao
import matplotlib.pyplot as plt
import numpy as np

# Para Jupyter: exibir graficos inline
%matplotlib inline

# Configuracoes opcionais (melhorar aparencia)
plt.rcParams['figure.dpi'] = 100  # Resolucao
plt.rcParams['font.size'] = 10     # Tamanho fonte

# Estilo (opcional)
# plt.style.use('seaborn-v0_8')  # Visual moderno
# plt.style.use('ggplot')        # Estilo R
# plt.style.use('default')       # Padrao matplotlib
```

Criar Primeira Figura

Python

```python
# Criar figura e axes
fig, ax = plt.subplots(figsize=(8, 5))
# Dados simples
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
# Plotar linha
ax.plot(x, y)
# Customizar
ax.set_xlabel('Tempo (dias)')
ax.set_ylabel('Crescimento (cm)')
ax.set_title('Crescimento de Plantas')
ax.grid(True, alpha=0.3)
# Exibir
plt.tight_layout()
plt.show()
```

Tipos de Gráficos para EDA

Três gráficos essenciais:

Histograma	**Boxplot**	**Bar Chart**
Mostra distribuição	Mostra quartis e outliers	Compara categorias
Uma variável numérica	Uma ou múltiplas variáveis	Variável categórica
Responde: ”Como os valores se distribuem?”	Responde: ”Qual o range e outliers?”	Responde: ”Qual categoria é maior?”

Nota Importante

Hoje: esses três. Próxima aula: scatter, heatmap, etc.

Histograma: O que é?

Histograma mostra a distribuição de dados numéricos

Como funciona:

1. Divide o range em ”bins” (intervalos)
2. Conta quantos valores caem em cada bin
3. Desenha uma barra para cada bin

Exemplo:

Alturas: 150, 160, 165, 170, 175, 180, 185

Bins: [150-160), [160-170), [170-180), [180-190)

Contagens: 2, 2, 2, 1

O que Histogramas Mostram?

Informações que você vê em um histograma:

- **Centro:** Onde a maioria dos valores está?
- **Espalhamento:** Quão ampla é a distribuição?
- **Forma:** Simétrica? Assimétrica?
- **Outliers:** Valores muito distantes?
- **Modas:** Um pico (unimodal) ou múltiplos picos?

Formas comuns:

- **Normal (sino):** Simétrica, um pico no centro
- **Assimétrica à direita:** Cauda longa à direita
- **Assimétrica à esquerda:** Cauda longa à esquerda
- **Uniforme:** Todas as barras mesma altura
- **Bimodal:** Dois picos distintos

Criar Histograma Básico

Python

```python
# Histograma simples
fig, ax = plt.subplots(figsize=(8, 5))

ax.hist(df['sepal length (cm)'],
        bins=20,           # Numero de intervalos
        edgecolor='black', # Borda das barras
        alpha=0.7)         # Transparencia

ax.set_xlabel('Sepal Length (cm)')
ax.set_ylabel('Frequency (count)')
ax.set_title('Distribuicao de Sepal Length')
ax.grid(True, alpha=0.3, axis='y')

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

Bins: O que São e Como Escolher

Bins = intervalos que agrupam valores

Exemplo com diferentes números de bins:

- **Poucos bins (5):** Perde detalhes, muito grosseiro
- **Bins moderados (20):** Bom equilíbrio
- **Muitos bins (100):** Muito detalhado, ruidoso

Como escolher?

- **Regra geral:** $\sqrt{n}$ onde n = número de observações
- Para Iris (150): $\sqrt{150} \approx 12$ bins
- Experimente: 10, 20, 30 e veja qual fica melhor

Nota Importante

Não existe número "certo" - depende dos dados e objetivo

Comparar Diferentes Números de Bins

</> Python

```python
# Comparar diferentes bins
fig, axes = plt.subplots(1, 3, figsize=(15, 4))

bins_list = [5, 20, 50]
col = 'sepal length (cm)'

for ax, bins in zip(axes, bins_list):
    ax.hist(df[col], bins=bins,
            edgecolor='black', alpha=0.7)
    ax.set_title(f'Bins = {bins}')
    ax.set_xlabel('Sepal Length (cm)')
    ax.set_ylabel('Frequency')

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

Interpretar Formas de Distribuição

Distribuição Simétrica (Normal):

Pico no centro, simétrica em ambos lados
Altura, temperatura, erros de medição

Assimétrica à Direita (Positiva):

Pico à esquerda, cauda longa à direita
Renda, preço de casas, tempo de espera

Assimétrica à Esquerda (Negativa):

Pico à direita, cauda longa à esquerda
Idade de morte, notas de provas fáceis

Histograma com Múltiplas Variáveis

Python

```python
# Histogramas de todas as variaveis numericas
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
axes = axes.flatten()
colunas = ['sepal length (cm)', 'sepal width (cm)',
           'petal length (cm)', 'petal width (cm)']

for ax, col in zip(axes, colunas):
    ax.hist(df[col], bins=20,
            edgecolor='black', alpha=0.7, color='steelblue')
    ax.set_xlabel(col)
    ax.set_ylabel('Frequency')
    ax.set_title(f'Distribuicao de {col}')
    ax.grid(True, alpha=0.3, axis='y')
plt.tight_layout()
plt.show()
```

Customizar Cores e Estilo

Python

```python
fig, ax = plt.subplots(figsize=(8, 5))
# Histograma customizado
ax.hist(df['petal length (cm)'],
        bins=25,
        color='lightcoral',      # Cor
        edgecolor='darkred',     # Cor da borda
        alpha=0.7,               # Transparencia
        linewidth=1.5)           # Largura da borda
ax.set_xlabel('Petal Length (cm)', fontsize=12, fontweight='bold')
ax.set_ylabel('Frequency', fontsize=12, fontweight='bold')
ax.set_title('Petal Length Distribution',
             fontsize=14, fontweight='bold')
ax.grid(True, alpha=0.3, linestyle='--', axis='y')
plt.tight_layout()
plt.show()
```

Exercício: Analisar Distribuições

✋ Exercício Prático

Crie histogramas e interprete:

1. Crie um histograma de `sepal width`
2. Use bins=20
3. Responda:
   - A distribuição é simétrica?
   - Onde está o pico?
   - Há outliers visíveis?
   - Quantos ”picos”você vê?
4. Experimente com bins=10 e bins=30
5. Qual número de bins revela melhor a estrutura?

Boxplot: O que é?

Boxplot visualiza quartis e outliers em um gráfico compacto

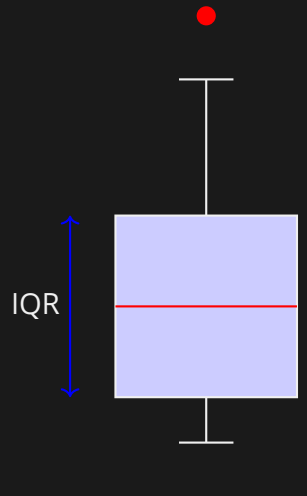
O que mostra:

- Mediana (linha no meio da caixa)
- Q1 e Q3 (extremos da caixa)
- Range dos dados ”normais”(whiskers/bigodes)
- Outliers (pontos individuais)

Vantagens:

- Mostra 5 estatísticas em um gráfico
- Identifica outliers visualmente
- Fácil comparar múltiplos grupos
- Compacto (ocupa pouco espaço)

Anatomia do Boxplot



Outlier

Máximo (dentro dos limites)

Q3 (75º percentil)

Mediana (Q2, 50º percentil)

Q1 (25º percentil)

Mínimo (dentro dos limites)

Outlier

Criar Boxplot Básico

Python

```python
# Boxplot simples
fig, ax = plt.subplots(figsize=(6, 8))
ax.boxplot(df['sepal length (cm)'])
ax.set_ylabel('Sepal Length (cm)')
ax.set_title('Boxplot de Sepal Length')
ax.grid(True, alpha=0.3, axis='y')
plt.tight_layout()
plt.show()

# Interpretacao visual:
# - Caixa mostra Q1 a Q3 (50% centrais)
# - Linha vermelha e a mediana
# - Linhas (whiskers) mostram range normal
# - Pontos isolados sao outliers
```

Interpretar Boxplot

O que você vê em um boxplot:

1. Posição da mediana:

- No centro da caixa $\rightarrow$ distribuição simétrica
- Perto de Q1 ou Q3 $\rightarrow$ distribuição assimétrica

2. Tamanho da caixa (IQR):

- Caixa pequena $\rightarrow$ dados concentrados
- Caixa grande $\rightarrow$ dados espalhados

3. Comprimento dos whiskers:

- Whiskers curtos $\rightarrow$ poucos valores extremos
- Whiskers longos $\rightarrow$ valores mais distribuídos

4. Outliers:

- Pontos além dos whiskers
- Valores anormalmente altos ou baixos

Boxplot *vs* Candlestick

⚠ Atenção

Boxplot NÃO É Candlestick!

Boxplot: Distribuição Estatística

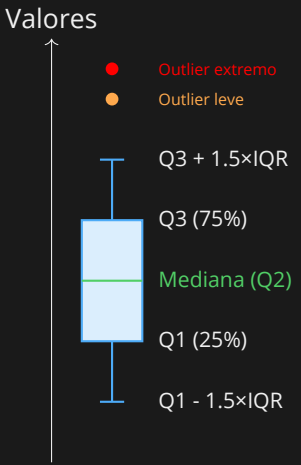
Objetivo: Resumir a distribuição de um dataset completo

Componentes:

- Caixa: Q1 → Q3 (50% central)
- Linha: Mediana (Q2)
- Whiskers: 1.5×IQR
- Pontos: Outliers

Uso:

- Comparar distribuições
- Identificar outliers
- Entender dispersão



Representa: distribuição de TODO o dataset

Candlestick: Movimento de Preços

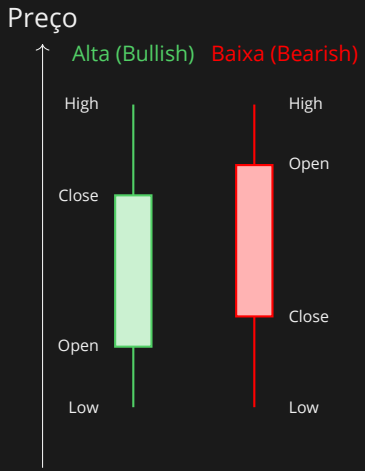
Objetivo: Mostrar movimento de preço em um período específico

Componentes:

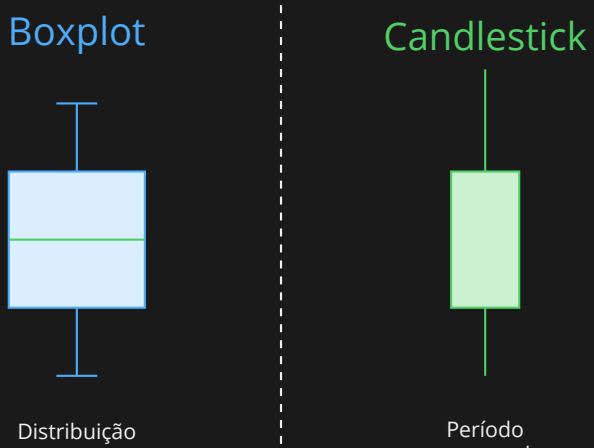
- Corpo verde: Preço subiu (Close > Open)
- Corpo vermelho: Preço caiu (Close < Open)
- Pavio superior: Máxima (High)
- Pavio inferior: Mínima (Low)

Uso:

- Análise técnica
- Identificar tendências
- Decisões de trading



Comparação: Boxplot *vs* Candlestick



Boxplot por Categoria

Python

```python
# Comparar especies com boxplots lado a lado
fig, ax = plt.subplots(figsize=(10, 6))
data_by_species = [
    df[df['species']=='setosa']['petal length (cm)'],
    df[df['species']=='versicolor']['petal length (cm)'],
    df[df['species']=='virginica']['petal length (cm)']
]
ax.boxplot(data_by_species,
           labels=['Setosa', 'Versicolor', 'Virginica'])
ax.set_ylabel('Petal Length (cm)')
ax.set_title('Petal Length por Especie')
ax.grid(True, alpha=0.3, axis='y')
plt.tight_layout()
plt.show()
```

Comparar Distribuições Visualmente

Boxplots lado a lado permitem comparações rápidas:

O que comparar:

- **Medianas:** Qual grupo tem valores mais altos?
- **IQR:** Qual grupo é mais homogêneo?
- **Overlap:** As caixas se sobrepõem? (grupos similares)
- **Outliers:** Qual grupo tem mais anomalias?
- **Simetria:** Mediana no centro ou deslocada?

Exemplo - Iris:

- Setosa: Pétalas muito menores (sem overlap)
- Versicolor: Tamanho intermediário
- Virginica: Pétalas maiores
- Conclusão: Petal length discrimina bem as espécies!

Boxplot vs Histograma: Quando Usar?

Use BOXPLOT quando:

- ✓ Comparar grupos
- ✓ Identificar outliers
- ✓ Mostrar quartis
- ✓ Espaço limitado
- ✓ Múltiplas variáveis

Exemplo:

- Salários por departamento
- Performance por região

Use HISTOGRAMA quando:

- ✓ Ver forma detalhada
- ✓ Identificar modas
- ✓ Uma variável
- ✓ Ver distribuição exata

Exemplo:

- Distribuição de idades
- Padrão de vendas

Nota Importante

Melhor: use AMBOS! Complementam-se.

Caso de Uso: Comparar Desempenho

Cenário: Comparar desempenho de 3 lojas

Dados de vendas diárias (R$):

- **Loja A:** Q1=5k, Mediana=7k, Q3=9k, sem outliers
- **Loja B:** Q1=4k, Mediana=6k, Q3=11k, 2 outliers (20k)
- **Loja C:** Q1=6k, Mediana=8k, Q3=10k, sem outliers

Insights do boxplot:

1. Loja C tem melhor mediana (mais consistente)
2. Loja B tem alta variabilidade (IQR grande)
3. Outliers da Loja B: dias de promoção?
4. Loja A é mais previsível (IQR menor)

Decisão: Investigar estratégias da Loja C e dias outliers da Loja B

Exercício: Boxplots do Iris

Exercício Prático

Explore com boxplots:

1. Crie boxplot de `sepal width` por espécie
2. Responda:
   - Qual espécie tem maior mediana?
   - Qual tem maior variabilidade (IQR)?
   - Há outliers? Em qual espécie?
   - As distribuições se sobrepõem?
3. Compare com `petal width`
4. Qual variável separa melhor as espécies?

Dica: Use o código do slide anterior como base

Bar Chart: O que é?

Bar chart (gráfico de barras) compara categorias

Quando usar:

- Variáveis categóricas (texto, não números)
- Comparar frequências ou valores entre grupos
- Mostrar contagens, totais, médias por categoria

Exemplos:

- Vendas por produto
- População por país
- Número de alunos por turma
- Frequência de espécies (Iris)

Bar Chart vs Histograma:

- Bar chart: categorias (discreto, espaçado)
- Histograma: valores contínuos (sem espaço entre barras)

Quando Usar: Comparar Categorias

Bar charts respondem:

- ”Qual categoria tem mais/menos?”
- ”Como as categorias se comparam?”
- ”Qual a distribuição de frequências?”

Exemplos práticos:

- **E-commerce:** Produtos mais vendidos
- **RH:** Funcionários por departamento
- **Marketing:** Canais de aquisição de clientes
- **Saúde:** Casos por região
- **Educação:** Distribuição de notas (A, B, C, D, F)

Nota Importante

Se você pode contar/categorizar, pode usar bar chart!

Criar Bar Chart Básico

Python

```python
# Contar frequencia de cada especie
counts = df['species'].value_counts()
# Criar bar chart
fig, ax = plt.subplots(figsize=(8, 6))
ax.bar(counts.index, counts.values,
       color='steelblue',
       edgecolor='black',
       alpha=0.7)
ax.set_xlabel('Species')
ax.set_ylabel('Count')
ax.set_title('Numero de Amostras por Especie')
ax.grid(True, alpha=0.3, axis='y')
plt.tight_layout()
plt.show()
```

Frequências com value_counts

Python

```python
# value_counts() conta ocorrencias
counts = df['species'].value_counts()
print(counts)
# Output:
# setosa        50
# versicolor    50
# virginica     50
props = df['species'].value_counts(normalize=True) # Com proporcoes (%)
print(props * 100)
# Output:
# setosa        33.33
# versicolor    33.33
# virginica     33.33

# Iris é perfeitamente balanceado!
```

Bar Chart Horizontal *vs* Vertical

Python

```python
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

counts = df['species'].value_counts()

# Vertical (padrao)
axes[0].bar(counts.index, counts.values, color='coral')
axes[0].set_title('Vertical Bar Chart')
axes[0].set_ylabel('Count')

# Horizontal (barh)
axes[1].barh(counts.index, counts.values, color='skyblue')
axes[1].set_title('Horizontal Bar Chart')
axes[1].set_xlabel('Count')
plt.tight_layout()
plt.show()
```

Bar Chart Horizontal *vs* Vertical

Nota Importante

Use horizontal quando labels são longos

Customização: Cores e Labels

Python

```python
counts = df['species'].value_counts()
fig, ax = plt.subplots(figsize=(8, 6))

# Cores diferentes por barra
colors = ['#FF6B6B', '#4ECDC4', '#45B7D1']
bars = ax.bar(counts.index, counts.values,
              color=colors,
              edgecolor='black',
              linewidth=1.5,
              alpha=0.8)
```

Customização: Cores e Labels (cont.)

Python

```python
# Customizar labels
ax.set_xlabel('Species', fontsize=12, fontweight='bold')
ax.set_ylabel('Count', fontsize=12, fontweight='bold')
ax.set_title('Iris Dataset - Species Distribution',
             fontsize=14, fontweight='bold')
ax.grid(True, alpha=0.3, axis='y')

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

Adicionar Valores nas Barras

Python

```python
counts = df['species'].value_counts()

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

bars = ax.bar(counts.index, counts.values,
              color='lightgreen', edgecolor='darkgreen')

# Adicionar texto em cima de cada barra
for i, (species, count) in enumerate(counts.items()):
    ax.text(i, count + 1,      # Posicao (x, y)
            str(count),         # Texto
            ha='center',        # Alinhamento horizontal
            fontsize=12,
            fontweight='bold')
```

Adicionar Valores nas Barras (cont.)

Python

```python
ax.set_ylabel('Count')
ax.set_title('Species Count (with values)')
ax.set_ylim(0, max(counts.values) + 10)  # Espaco para texto

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

Bar Chart de Médias por Grupo

Python

```python
# Calcular media de petal length por especie
means = df.groupby('species')['petal length (cm)'].mean()

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

ax.bar(means.index, means.values,
       color='orchid', edgecolor='purple', alpha=0.7)

# Adicionar valores
for i, (species, mean) in enumerate(means.items()):
    ax.text(i, mean + 0.1, f'{mean:.2f} cm',
            ha='center', fontweight='bold')
```

Bar Chart de Médias por Grupo (cont.)

Python

```python
ax.set_ylabel('Average Petal Length (cm)')
ax.set_title('Average Petal Length por Especie')
ax.grid(True, alpha=0.3, axis='y')

plt.tight_layout()
plt.show()

```

Exercício: Análise Categórica

✋ Exercício Prático

Crie análises categóricas:

1. Crie uma nova categoria baseada em `sepal length`:

```python
# Classificar em pequeno/medio/grande
df['size_category'] = pd.cut(
    df['sepal length (cm)'],
    bins=[0, 5.5, 6.5, 10],
    labels=['Small', 'Medium', 'Large']
)
```

2. Crie bar chart mostrando frequência de cada categoria
3. Adicione os valores nas barras
4. Interprete: qual categoria é mais comum?

Checklist de EDA Básica

Sempre execute estas etapas:

1. ☐ **Carregar e visualizar**
   - head(), tail(), sample()
2. ☐ **Dimensões e tipos**
   - shape, info(), dtypes
3. ☐ **Valores faltantes**
   - isnull().sum()
4. ☐ **Estatísticas descritivas**
   - describe(), mean(), median(), std()
5. ☐ **Visualizações básicas**
   - Histogramas, boxplots
6. ☐ **Análise por grupos**
   - groupby() + estatísticas
7. ☐ **Documentar insights**

Erros Comuns e Como Evitar

Erros frequentes em EDA:

1. **Não verificar valores faltantes**
   - ✗ Assumir que dados estão completos
   - ✓ Sempre use isnull().sum()
2. **Confiar apenas em média**
   - ✗ Ignorar mediana e outliers
   - ✓ Use média E mediana, verifique outliers
3. **Não visualizar**
   - ✗ Só olhar números
   - ✓ Sempre criar gráficos
4. **Escolher bins inadequados**
   - ✗ Usar sempre mesmo número
   - ✓ Experimentar diferentes valores
5. **Não documentar insights**
   - ✗ Só gerar gráficos
   - ✓ Anotar o que você descobriu

Recap: Estatísticas Fundamentais

O que aprendemos hoje:

Tendência Central:

- Média: valor ”típico”(sensível a outliers)
- Mediana: valor do meio (robusta)
- Moda: valor mais frequente (categóricas)

Dispersão:

- Range: max - min (simples)
- Desvio padrão: dispersão ”típica”(use este!)
- IQR: largura dos 50% centrais

Quartis:

- Q1, Q2 (mediana), Q3: dividem em 4 partes
- Outliers: $< Q1 - 1.5 \times IQR$ ou $> Q3 + 1.5 \times IQR$

Recap: Gráficos Básicos

Três gráficos essenciais:

Gráfico	Uso	Mostra
Histograma	Distribuição	Forma, centro, spread
Boxplot	Quartis + outliers	Q1, Q2, Q3, anomalias
Bar Chart	Categorias	Frequências, comparações

Quando usar cada um:

- **Histograma:** Ver distribuição detalhada de 1 variável
- **Boxplot:** Comparar grupos, identificar outliers
- **Bar Chart:** Comparar categorias

Próxima aula: Scatter plots, correlação, heatmaps (Seaborn)

Preview Aula 08: EDA Parte 2

Na quinta-feira (30/10) veremos:

- **Correlação:** Relações entre variáveis
- **Scatter plots:** Visualizar correlações
- **Seaborn:** Visualizações mais bonitas e fáceis
- **Heatmaps:** Matriz de correlação visual
- **Pairplot:** Ver tudo de uma vez
- **Análise multivariada:** Múltiplas variáveis juntas
- **Storytelling:** Comunicar insights

Preparação:

- Revisar conceitos de hoje
- Praticar com Iris
- Pensar em perguntas sobre relações entre variáveis

Dicas para a Entrega Quinzenal

Segunda entrega (sexta-feira 31/10):

O que incluir na EDA:

1. Exploração inicial (shape, info, describe)
2. Valores faltantes (identificar e tratar)
3. Estatísticas por grupo
4. No mínimo 5 visualizações:
   - Histogramas
   - Boxplots
   - Bar charts
   - (Quinta-feira: scatter, heatmaps)
5. Documentar insights em markdown

Dica: Use este notebook como template!

Recursos para Estudo

Documentação:

- Pandas: `https://pandas.pydata.org/docs/`
- Matplotlib: `https://matplotlib.org/`
- Matplotlib Gallery: exemplos visuais prontos

Prática:

- Kaggle: Iris dataset notebooks
- UCI ML Repository: outros datasets
- Pratique com dados do seu interesse

Conceitos:

- Khan Academy: Statistics
- StatQuest (YouTube): conceitos visuais

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: EDA Parte 2 (Seaborn e Correlações)
Quinta-feira, 30/10

Dúvidas: via Moodle ou agora na prática