

```
import numpy as np
```

```
dados = np.array([85, 92, 78, 94, 89, 110, 45, 75, 82, 99])
```

```
# Cálculo da amplitude  
amplitude = np.max(dados) - np.min(dados)  
print("Amplitude dos Dados:", amplitude)
```

```
↩↪ Amplitude dos Dados: 65
```

```
# Cálculo da média  
media = np.mean(dados)
```

```
# Cálculo da variância  
variancia = np.mean((dados - media) ** 2)  
print("Variância dos Dados:", variancia)
```

```
↩↪ Variância dos Dados: 272.49
```

```
# Cálculo do desvio padrão  
desvio_padrao = np.sqrt(variancia)  
print("Desvio Padrão dos Dados:", desvio_padrao)
```

```
↩↪ Desvio Padrão dos Dados: 16.5072711251739
```

```
# Cálculo do primeiro quartil (Q1)  
q1 = np.percentile(dados, 25)
```

```
# Cálculo do terceiro quartil (Q3)  
q3 = np.percentile(dados, 75)
```

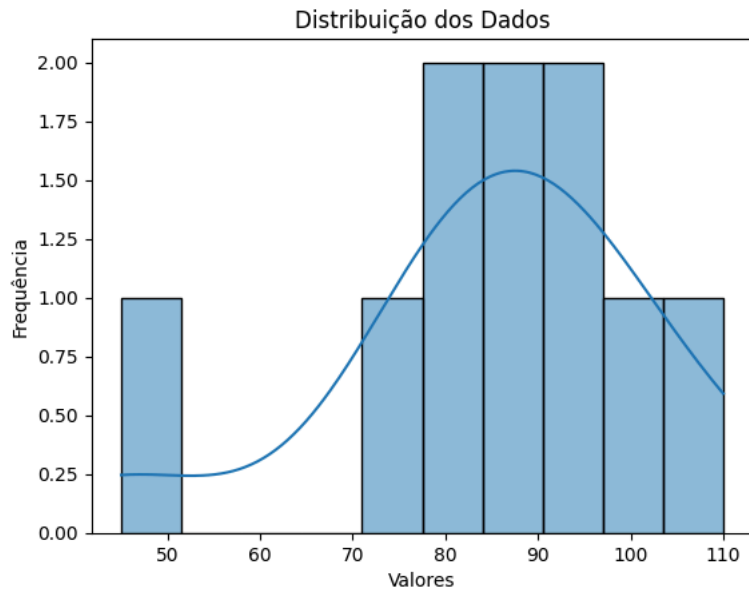
```
# Cálculo do IQR  
iqr = q3 - q1  
print("Primeiro Quartil (Q1):", q1)  
print("Terceiro Quartil (Q3):", q3)  
print("Intervalo Interquartil (IQR):", iqr)
```

```
↩↪ Primeiro Quartil (Q1): 79.0  
    Terceiro Quartil (Q3): 93.5  
    Intervalo Interquartil (IQR): 14.5
```

▼ prática com meus proprios dados

```
import matplotlib.pyplot as plt  
import seaborn as sns
```

```
sns.histplot(dados, bins=10, kde=True)  
plt.title('Distribuição dos Dados')  
plt.xlabel('Valores')  
plt.ylabel('Frequência')  
plt.show()
```



```
def calc_estatisticas(dados):
    amplitude = np.max(dados) - np.min(dados)
    media = np.mean(dados)
    variancia = np.mean((dados - media) ** 2)
    desvio_padrao = np.sqrt(variancia)
    q1 = np.percentile(dados, 25)
    q3 = np.percentile(dados, 75)
    iqr = q3 - q1

    return {
        'Amplitude': amplitude,
        'Variância': variancia,
        'Desvio Padrão': desvio_padrao,
        'Q1': q1,
        'Q3': q3,
        'IQR': iqr
    }

resultados = calc_estatisticas(dados)
for chave, valor in resultados.items():
    print(f"{chave}: {valor}")
```



```
Amplitude: 65
Variância: 272.49
Desvio Padrão: 16.5072711251739
Q1: 79.0
Q3: 93.5
IQR: 14.5
```

↘ maematica pratica

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.dates as mdates

url = "https://raw.githubusercontent.com/Professor-Leonardo/temperatura/main/DataCidadeEstadoTemperatura.csv"
dados_temperatura = pd.read_csv(url, sep=';', encoding='ISO-8859-1')

# Converter a coluna 'Temperatura' para numérica, removendo vírgulas e convertendo para float
dados_temperatura['Temperatura'] = pd.to_numeric(dados_temperatura['Temperatura'].str.replace(',', '.'))

# Converter a coluna 'Data' para o tipo datetime
dados_temperatura['Data'] = pd.to_datetime(dados_temperatura['Data'], format='%d/%m/%Y')

# Passo 3: Análise Inicial dos Dados
print("### Primeiras linhas do conjunto de dados ###")
print(dados_temperatura.head())

print("\n### Informações gerais sobre o conjunto de dados ###")
print(dados_temperatura.info())
```

```
print("\n### Resumo estatístico das variáveis numéricas ###")
print(dados_temperatura.describe())
```

```
# Estatísticas descritivas
media_temperatura = dados_temperatura['Temperatura'].mean()
mediana_temperatura = dados_temperatura['Temperatura'].median()
variância_temperatura = dados_temperatura['Temperatura'].var()
desvio_padrao_temperatura = dados_temperatura['Temperatura'].std()
Q1 = dados_temperatura['Temperatura'].quantile(0.25)
Q3 = dados_temperatura['Temperatura'].quantile(0.75)
IQR_temperatura = Q3 - Q1
```

```
# Criar tabela com estatísticas
estatisticas = {
    "Média": media_temperatura,
    "Mediana": mediana_temperatura,
    "Variância": variância_temperatura,
    "Desvio Padrão": desvio_padrao_temperatura,
    "Q1": Q1,
    "Q3": Q3,
    "IQR": IQR_temperatura
}
```

```
df_estatisticas = pd.DataFrame([estatisticas], index=["Temperatura"])
```

```
# Exibir a tabela
print("\nTabela de Estatísticas Descritivas:")
print(df_estatisticas)
```

```
## Primeiras linhas do conjunto de dados ##
  Data      Cidade Estado  Temperatura
0 2023-01-01    Campinas  SP         30.2
1 2023-01-02    Campinas  SP         30.5
2 2023-01-03    Campinas  SP         29.9
3 2023-01-01  Ribeirao Preto  SP         32.8
4 2023-01-02  Ribeirao Preto  SP         33.2
```

```
## Informações gerais sobre o conjunto de dados ##
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312 entries, 0 to 311
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Data        312 non-null    datetime64[ns]
1   Cidade      312 non-null    object
2   Estado      312 non-null    object
3   Temperatura 312 non-null    float64
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 9.9+ KB
None
```

```
## Resumo estatístico das variáveis numéricas ##
      Data  Temperatura
count      312    312.000000
mean  2023-02-08 21:04:36.923076864    28.240385
min      2023-01-01 00:00:00    18.400000
25%      2023-01-25 00:00:00    25.550000
50%      2023-02-07 00:00:00    28.850000
75%      2023-02-26 06:00:00    31.200000
max      2023-03-24 00:00:00    34.400000
std              NaN         3.928464
```

```
Tabela de Estatísticas Descritivas:
      Média  Mediana  Variância  Desvio Padrão    Q1    Q3    IQR
Temperatura  28.240385    28.85  15.432833    3.928464  25.55  31.2  5.65
```

```
# Box plot das temperaturas por Estado
```

```
plt.figure(figsize=(12, 7))
sns.boxplot(data=dados_temperatura, x='Estado', y='Temperatura', palette="Set2", linewidth=2.5)
plt.title('Box Plot das Temperaturas por Estado', fontsize=16, fontweight='bold')
plt.xlabel('Estado', fontsize=12)
plt.ylabel('Temperatura (°C)', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True, linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()
```

```
# Explicação:
```

```
# Este gráfico mostra a distribuição das temperaturas para cada estado. O box plot exibe a mediana,
# quartis (Q1 e Q3), e possíveis outliers. O gráfico é útil para visualizar as diferenças de temperatura entre estados
# e também para identificar valores atípicos em determinados estados.
```

```
# histograma de temperatura por cidade
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(14, 7))

# Usar histplot para criar o histograma
sns.histplot(data=dados_temperatura, x='Temperatura', hue='Cidade', multiple='stack', palette="husl", bins=20)
# Explicação:
# O histograma exibe a distribuição das temperaturas em diferentes cidades. As barras empilhadas indicam
# quantas vezes uma determinada faixa de temperatura foi registrada em cada cidade. É possível visualizar
# quais cidades têm maiores ou menores concentrações em faixas específicas de temperatura.

# Configurações de formatação do gráfico
plt.title('Distribuição das Temperaturas por Cidade', fontsize=16, fontweight='bold')
plt.xlabel('Temperatura (°C)', fontsize=12)
plt.ylabel('Frequência', fontsize=12)

plt.grid(True, linestyle='--', linewidth=0.5)
plt.tight_layout()
plt.show()

# Gráfico de dispersão da temperatura por cidade
plt.figure(figsize=(12, 7))
sns.scatterplot(data=dados_temperatura, x='Cidade', y='Temperatura', hue='Estado', palette='viridis', s=100)
plt.title('Dispersão das Temperaturas por Cidade', fontsize=16)
plt.xlabel('Cidade', fontsize=12)
plt.ylabel('Temperatura (°C)', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
# Explicação:
# Este gráfico de dispersão mostra as temperaturas para cada cidade, com a cor indicando o estado
# ao qual a cidade pertence. Isso facilita a identificação de padrões geográficos em relação à variação
# de temperatura e também ajuda a visualizar quais cidades têm variações maiores ou menores.

# Histograma da distribuição de temperatura global
plt.figure(figsize=(12, 7))
sns.histplot(dados_temperatura['Temperatura'], bins=30, kde=True, color='blue')
plt.title('Histograma da Distribuição Global das Temperaturas', fontsize=16)
plt.xlabel('Temperatura (°C)', fontsize=12)
plt.ylabel('Frequência', fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()
# Explicação:
# O histograma exibe a distribuição das temperaturas em diferentes cidades. As barras empilhadas indicam
# quantas vezes uma determinada faixa de temperatura foi registrada em cada cidade. É possível visualizar
# quais cidades têm maiores ou menores concentrações em faixas específicas de temperatura.

# Graficos - Cidades
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.dates as mdates

# Iterar sobre as cidades e criar gráficos separados
cidades = dados_temperatura['Cidade'].unique()

for cidade in cidades:
    dados_cidade = dados_temperatura[dados_temperatura['Cidade'] == cidade]

    plt.figure(figsize=(10, 6))

    # Gráfico de linha sem suavização
    sns.lineplot(data=dados_cidade, x='Data', y='Temperatura', color='blue', linewidth=2)

    # Marcar o ponto máximo
    max_temp = dados_cidade['Temperatura'].max()
    max_temp_date = dados_cidade[dados_cidade['Temperatura'] == max_temp]['Data'].values[0]
    plt.text(max_temp_date, max_temp, f'{max_temp:.1f}°C', color='black', fontsize=10, verticalalignment='bottom', horizontalalignment='right')
    plt.scatter(max_temp_date, max_temp, color='red', s=100, zorder=5, edgecolor='black')


    # Marcar o ponto mínimo
    min_temp = dados_cidade['Temperatura'].min()
    min_temp_date = dados_cidade[dados_cidade['Temperatura'] == min_temp]['Data'].values[0]
    plt.text(min_temp_date, min_temp, f'{min_temp:.1f}°C', color='black', fontsize=10, verticalalignment='top', horizontalalignment='right')
    plt.scatter(min_temp_date, min_temp, color='blue', s=100, zorder=5, edgecolor='black')

    # Configurações de formatação do gráfico
    plt.title(f'Temperatura Diária em {cidade}', fontsize=16, fontweight='bold')
    plt.xlabel('Data de Observação', fontsize=12)
```

```
plt.ylabel('Temperatura (°C)', fontsize=12)
plt.xticks(rotation=45, fontsize=10)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%m/%Y')) # Formato de mês/ano

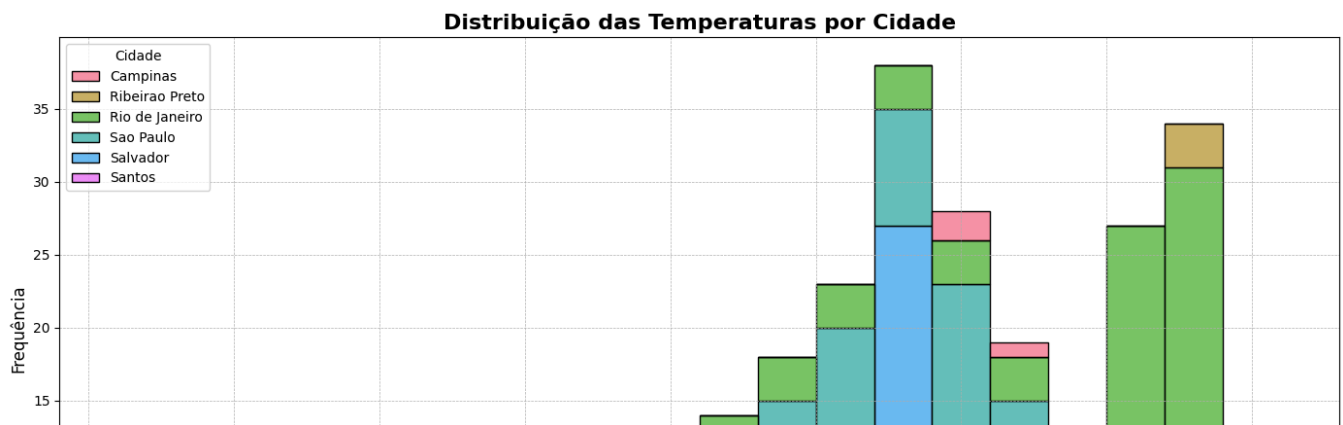
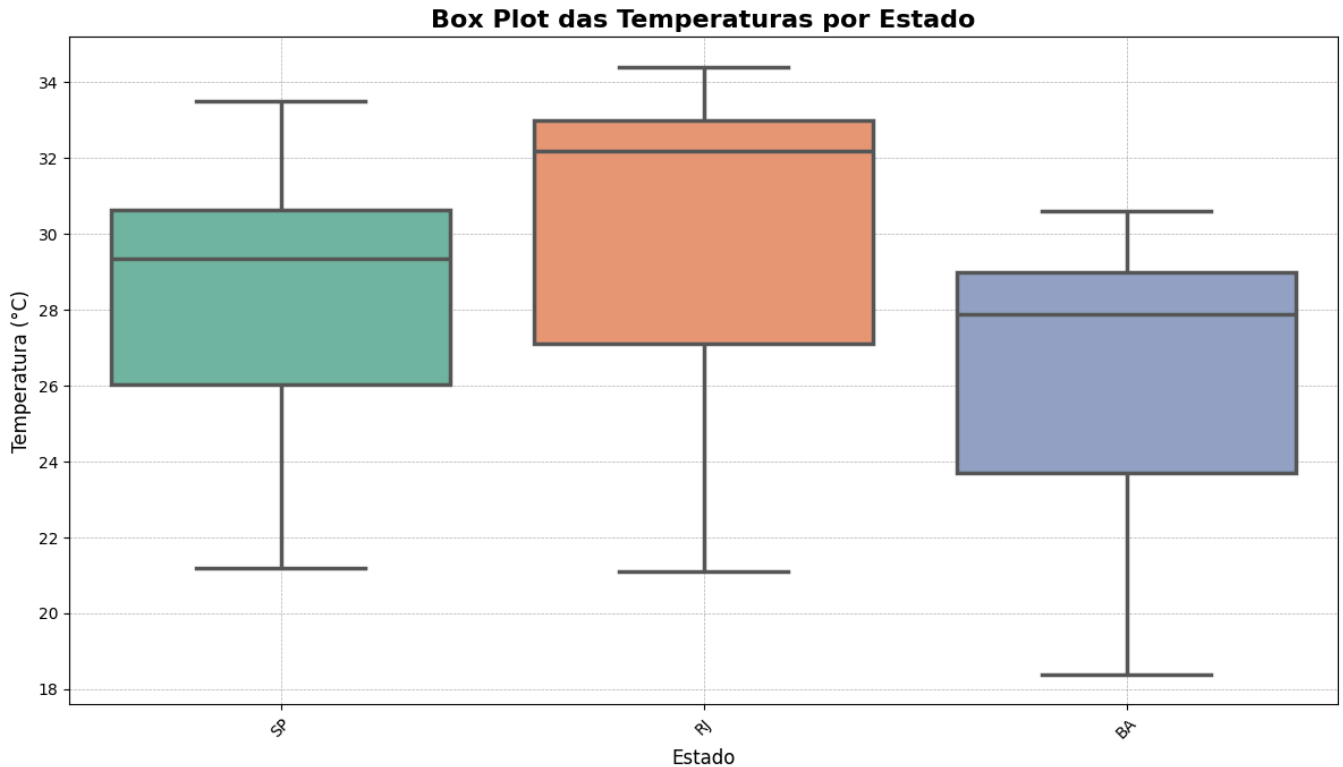
# Adicionar grade ao gráfico
plt.grid(True, which='both', linestyle='--', linewidth=0.7)

plt.tight_layout() # Ajustar automaticamente o layout
plt.show()
```

 <ipython-input-20-39711a521679>:4: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

```
sns.boxplot(data=dados_temperatura, x='Estado', y='Temperatura', palette="Set2", linewidth=2.5)
```



```
#Estatísticas descritivas das temperaturas
media_temperatura = dados_temperatura['Temperatura'].mean()
mediana_temperatura = dados_temperatura['Temperatura'].median()
variancia_temperatura = dados_temperatura['Temperatura'].var()
desvio_padrao_temperatura = dados_temperatura['Temperatura'].std()
Q1 = dados_temperatura['Temperatura'].quantile(0.25)
Q3 = dados_temperatura['Temperatura'].quantile(0.75)
IQR_temperatura = Q3 - Q1
```

```
print("\n### Estatísticas das Temperaturas ###")
print("Média:", media_temperatura)
print("Mediana:", mediana_temperatura)
print("Variância:", variancia_temperatura)
print("Desvio Padrão:", desvio_padrao_temperatura)
print("Intervalo Interquartil (IQR):", IQR_temperatura)
```

 ### Estatísticas das Temperaturas ###

Média: 28.240384615384617

Mediana: 28.85
Variância: 15.432833292109821
Desvio Padrão: 3.928464495467549